Synthesizing and Editing Photo-realistic Visual Objects

Daniyar Turmukhambetov

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

October 20, 2016

I, Daniyar Turmukhambetov, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The goal of this thesis is to investigate novel methods of synthesizing new images of a deformable visual object using a collection of images of the object. We investigate both parametric and non-parametric methods as well as a combination of the two for the problem of image synthesis. Our main focus is complex visual objects, specifically deformable visual objects and visual objects with varying numbers of visible parts, which are challenging for existing techniques.

We first introduce a system for interactive sketch-driven image synthesis that allows the user to draw ellipses and outlines in order to sketch a rough shape of animals as a constraint to the synthesized image. This system interactively provides feedback in the form of ellipse and contour suggestions to the partial sketch of the user, by leveraging a parametric representation of ellipse positions and object outlines. The user's sketch guides the non-parametric synthesis algorithm that blends patches from two exemplar images in a coarse-to-fine fashion to create a final image that satisfies the user's constraints. We evaluate the method and synthesized images through two user studies.

Instead of non-parametric blending of patches from just a few, weakly aligned, images, a parametric model of the appearance is more desirable, because it captures the appearance information that is shared between all images of the dataset. Hence, we propose Context-Conditioned Component Analysis (C-CCA), a probabilistic generative parametric model that describes images with a linear combination of basis functions. The basis functions are evaluated for each pixel, where the arguments of the functions are context vectors computed from local shape information. This allows us to model the appearance of deformable objects whilst also dealing with complex occlusions and varying numbers of parts. We demonstrate C-CCA on appearance transfer and structured inpainting tasks. We evaluate the quality of the appearance reconstruction with numerical and perceptual metrics.

C-CCA can be directly sampled to generate novel, globally-coherent images. Unfortunately, these samples lack high-frequency details due to dimensionality reduction and misalignment. Hence, we develop a non-parametric model that enhances the samples of C-CCA with locally-coherent, high-frequency details. Thus, we can synthesize highly-detailed photo-realistic images of deformable visual objects by combining the global parameteric model and local non-parametric model. The non-parametric model stitches patches that match the sample drawn from C-CCA but also contain high-frequency details. The patch correspondences are efficiently searched in all images of the dataset with allowed rotation and scale variations. We show and analyze the results of the combined method on the datasets of horse and elephant images.

Acknowledgements

I'm incredibly grateful to my supervisors Jan Kautz and Simon J.D. Prince and my collaborator Neill D.F. Campbell for the guidance, support, encouragement and wisdom.

I'm especially thankful to Gabriel Brostow, for supervising my masters thesis, encouraging me to pursue a PhD degree and providing advice during my research.

I'm filled with gratitude to Dan B Goldman and Adobe Systems Inc. for the wonderful internship at Adobe and the long collaboration on my research.

Many thanks to everyone in the VECG group for making my time at UCL memorable, enjoyable and productive.

I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for funding the research grant(EP/I031170/1) and UCL Faculty of Engineering for providing Postgraduate Research Scholarship.

Finally, I would like to thank my family for all their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"I don't know where the artificial stops and the real starts."

—Andy Warhol

Қазақтың халық жұмбағы:
Терезені өрнектеп,
Әдемі сурет салыпты.
Кім екенін табайық,
Көрінбей кетіп қалыпты.

(ɘʁ∀)

In this thesis we investigate novel methods of synthesizing photo-realistic images of deformable objects. Synthesis of novel photo-realistic images has many current and potential applications such as: generating content for virtual worlds and games, software for assisting in conceptual design of new products for manufacturing, clip-art-like image generation with richer control over the result, photo and video editing software, *etc*...

For creating realistic images, there are three main families of approaches: Computer Graphics Rendering, Non-Parametric Image Synthesis (*i.e.*, Image-Based Rendering) and Generative Parametric Models.

Computer Graphics Rendering Computer Graphics Rendering is a mature research area that can generate photo-realistic images with near-perfect results. However, the current pipeline requires a lot of user input, such as specifying object materials, 3D geometry, lighting, camera properties, volume properties, *etc*. The currently available tools for 3D rendering and modeling are not easy to use for novices and require substantial training. Achieving high-quality results is often a time-consuming procedure, even for experienced and trained computer graphics professionals. Moreover, after all the necessary properties have been specified, modifying the result remains challenging and time-consuming, as it

is hard to edit the resulting digital images at anything other than the low level of geometry and materials at which they are specified. Changes at such low level may have unpredictable effects on the resulting image (*e.g.* changing the glossiness of a single object in the scene may change overall brightness of the rendered image). Finally, editing existing photographs with these rendering techniques is even more difficult, as the geometry, appearance and the lighting setup of the photograph have to be replicated in a virtual scene.

Non-parametric Image Synthesis There are other works which have a principally different approach in generating photo-realistic images that work in 2D image space. The image synthesis is done non-parametrically by reusing regions of images from a library of exemplars. For example, there are methods that address the problem of texture synthesis, where a small image of a texture is used as input for generating a larger image with the same texture (see section 2.2.1). This is done by copying and pasting pixels or patches from the input image into the output image while preserving local coherence. For the input images of homogeneous textures with locally similar structure, such methods produce high-quality results (*e.g.* Figure 1.1 (A)). Unfortunately, these methods lack the notion of global coherence, thus, they are generally not applicable for images with non-homogeneous (*i.e.* varying) textures, such as faces, buildings, animals, *etc.* (an example is shown in Figure 1.1 (B)). Nevertheless, these methods work well in tasks of image inpainting, image blending, painting-with-numbers, *etc.* (see section 2.2). Another non-parametric approach of generating photo-realistic images is to edit existing images by blending in new visual objects (see section 2.2.3). However, these methods cannot generate new instances of visual objects. For example, one can edit an image of a street by copy-pasting (*i.e.* compositing) cars from other images, but it is not possible to generate a novel car, or modify the attributes of a certain car.

Generative Parametric Models Finally, the third principally different way of generating novel images of visual objects is to sample a generative probabilistic model of an object fitted to the dataset of images. So, one can randomly sample the parameter space and use the model to reconstruct the corresponding images in the image space, which produces globally-coherent results (see Figure 1.2). Although typical images have the number of dimensions in thousands or millions, depending on the resolution, the natural images have structure [2, 3], which implies that natural images lie on a low-dimensional manifold. So, dimensionality reduction techniques are applied to build a low-

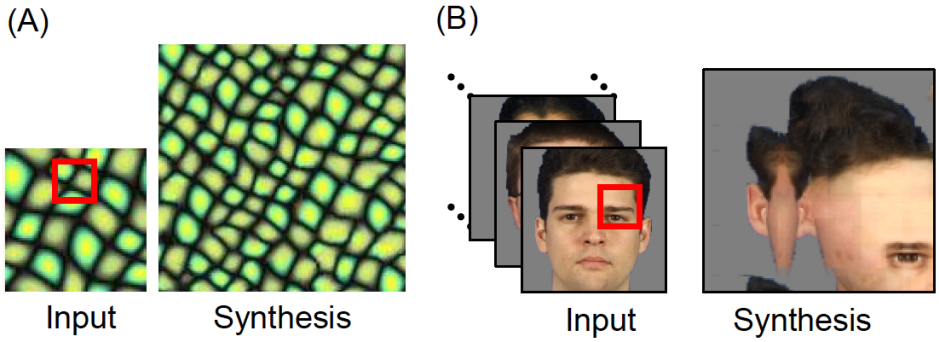


Figure 1.1: (A) Texture synthesis from input exemplar. (B) Generating faces with the texture synthesis algorithm. The synthesized image lacks global coherence. [Source: [1]].

dimensional representation of the images, which preserves as much information as possible. Because of the dimensionality reduction and the inherent properties of the models (such as Gaussian noise assumption), the reconstructions lack high-frequency details, thus, they are not photo-realistic. Alternatively, one can avoid dimensionality reduction altogether, but this will produce a model that generates blends or morphings of two or more exemplars from a dataset. However, choosing the exemplars and ensuring global coherence is not trivial, for example, blending a young female face with an elderly male face is unlikely to produce a photo-realistic result.

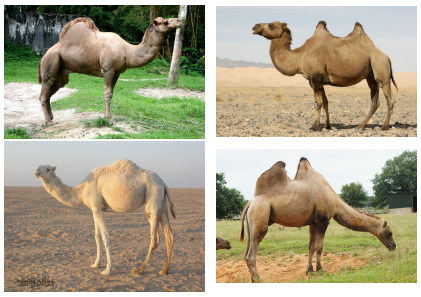
Figure 1.2: Random samples of the factor analysis model trained on images of faces. [Source: [1]].

Furthermore, these models require registration of the data (*i.e.* all images of the object are warped to align with one-another). The alignment is well-defined for visual objects with small deformations and fixed structure, *e.g.* faces with neutral expression in a frontal pose, as in Figure 1.3 (a). However, *highly-deformable* objects, such as animals, can be in poses with self-occlusions which

(a) Rigid and Fixed structure

(b) Highly-Deformable and Fixed structure

(c) Rigid and Non-structured



(d) Highly-deformable and Non-structured

Figure 1.3: Examples of datasets of rigid and highly-deformable visual objects with fixed and varying structure. (a) Rigid and Fixed structure. [Source: [6]]. (b) Highly-Deformable and Fixed structure. [Source: [219]]. (c) Rigid and Non-structured. [Source: [240]]. (d) Highly-deformable and Non-structured. [Sources to unmodified, original images left to right, top to bottom: "*Camel aka Unta*" by *Phalinn Ooi* / CC BY 2.0; "*Camelus bactrianus - Camel*" by *Jerrold Bennett* / CC BY-NC-ND 2.0; "*Desert Camel*" by *Nerissa Alford* / CC BY-NC-ND 2.0; "*Bactrian Camel*" by *Tim Ellis* / CC BY-NC 2.0].

makes the problem of alignment very difficult. For example, it is unclear how can one warp an image of a standing horse with only two legs visible to a galloping horse with all four legs visible, as in Figure 1.3 (b). Consequent shortcoming is inability to transfer appearance across different poses. In addition, current parametric models require that the visual object has *fixed structure*, where

elements of the visual object always appear in the same quantity (for example, frontal images of faces always have two eyes that are always visible). On the other hand, it is unclear how *non-structured* objects can be aligned in image space. For instance, images of facades of buildings may have different numbers of windows, therefore it is impossible to warp these images of facades to a common template as exemplified in Figure 1.3 (c). Non-structured visual objects can also be highly-deformable, *e.g.*, camels with one or two humps in Figure 1.3 (d).

(a) Images of the training set

(b) Reconstructions of the images of the training set

(c) Ten random samples of the model

Figure 1.4: PPCA was trained on 200 images of horses [219] of size 75x75 with 20 components. The reconstructions of the training set in (b) and samples in (c) demonstrate PPCA's quality of modeling the appearance of the horses.

This thesis investigates non-parametric methods, parametric methods and combinations of the two for generating images, with the focus on synthesis of *deformable* objects. Although existing methods synthesize high-quality results

(a) Images of the training set

(b) Reconstructions of the images of the training set

(c) Ten random samples of the model

Figure 1.5: PPCA was trained on 200 images of elephants of size 68x68 with 20 components. The reconstructions of the training set in (b) and samples in (c) demonstrate PPCA's quality of modeling the appearance of the elephants.

for visual objects that are aligned and have fixed structure, these methods are not applicable for more complex, deformable visual objects. For example, Risser *et al.* [4] produce image hybrids by preserving the structure of input exemplars (see Figure 1.6 (b)). Mohammed *et al.* [1] learn to synthesize high-quality images of faces from a dataset of aligned face images, with a combination of parametric and non-parametric methods. Mohammed *et al.* use Probabilistic Principal Component Analysis [5] as the parametric model, which performs poorly for unaligned images and produces samples that are not

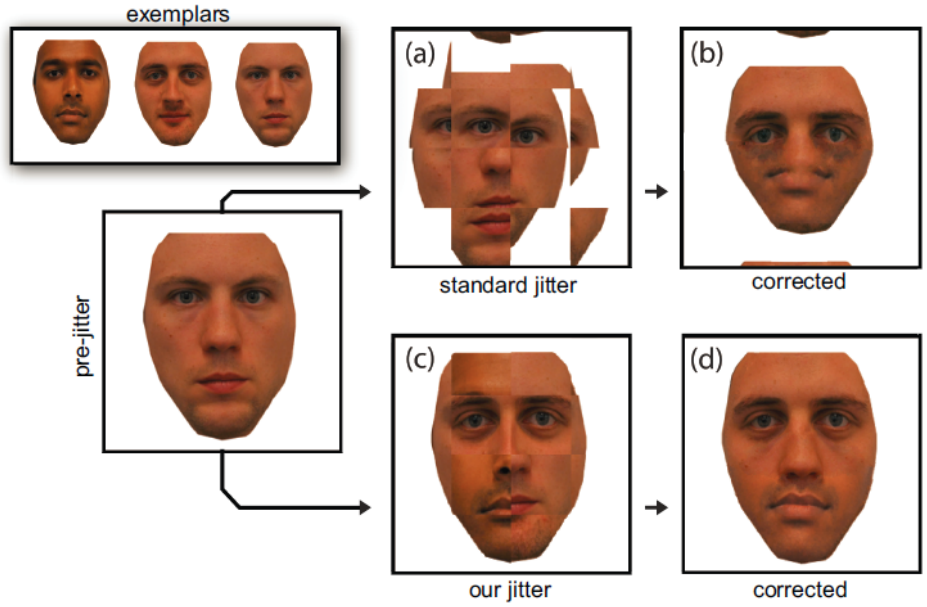


Figure 1.6: Structure preserving jitter of Risser *et al*. [Source: [4]].

globally coherent (see Figure 1.4 and 1.5). We address the problem of image synthesis of complex visual objects by exploiting a structure map. A structure map is an image of discrete labels where each label indicates the part of the object that is present at that position (see Figure 1.7).

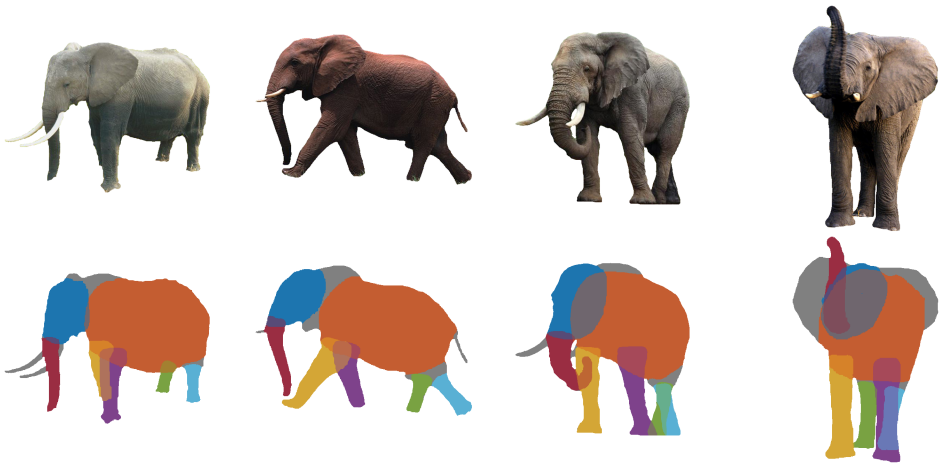


Figure 1.7: Images of elephants and visualization of the corresponding structure maps. The grey color in the structure map represents the pixels that belong to the elephants. The overlaid colors encode the labelings of the trunk, head, torso and 4 legs.

We show how non-parametric image synthesis can produce high-quality results for deformable, non-aligned visual objects by guiding the synthesis

process with the user's sketch input. Furthermore, we demonstrate that the combined method can generate images of highly-deformable, non-aligned visual objects by formulating a new parametric, globally-coherent, probabilistic model that allows structure and pose variation. Then, we can synthesize novel images of that class of objects by drawing samples from this parametric model. Moreover, this model can be fit to an existing image for initialization or to a partial view of the image for inpainting.

1.1 Objectives

The aim of this thesis is to investigate synthesis and editing of photo-realistic images of deformable visual objects using a collection of images of different instances of the class of the visual object. This is a challening problem with a number of different desired properties.

- Firstly, we would like a data-driven system that can extract information from the data which later can assist a novice user to create new images. This system has to be interactive, capable of providing helpful feedback to the user, so that the final result suits the user's constraints and looks realistic. For example, the system can suggest to the user which of the specified constraints are not realistic (*e.g.*, the head of the cat is too big compared to the torso) and guide the user towards more realistic constraints, which is helpful if the user is not artistically inclined.

- Secondly, we would like to learn a generative model from a dataset of images of a highly-deformable, non-structured visual object. This generative model should have following properties:

  - The model has to be able to generate novel photo-realistic images of the visual object. This could be completely automatic, or with certain user constraints.

  - The model should also be capable of initializing from an existing image rather than synthesizing a new one. This is desirable for tasks such as appearance transfer: where the user has an image of an object with a suitable appearance, but in a different, or partially occluded pose. By initializing the model on the existing image, the representation of the appearance can be modified; transferred to an object in a different pose; or used to inpaint an occluded region of the image. As the model learns the subspace of appearances in

an unsupervised fashion, the goal of the appearance transfer and inpainting tasks is to generate a plausible result. This is different from accuracy-driven appearance transfer methods that are provided with images of the same object in different poses.

– The model has to allow control over the result such as specifying constraints on image level or semantic level. For example, the user could control the synthesis by drawing contours in some parts of the image or by specifying color palette. Alternatively, the user could adjust a few parameters such that the changes have intuitive and consistent effect on the result.

1.2 Challenges

This work faces both practical and scientific challenges:

Data Collection While there are different corpora of face images ([6, 7]) captured in controlled environments with perfect fiducial point annotations, different lighting conditions, segmentation masks, *etc.*, there are very few publicly available datasets for other visual objects with sufficient number of images. A lot of datasets were compiled for image classification and object detection problems, but, as a result, the images don't have segmentation masks and may have occlusions. Annotating correspondences, segmenting the visual object from the background and labeling semantic parts to create a structure map is necessary to learn a model that can condition the appearance on the pose and the structure. However, compiling such a dataset by an individual would still result in a relatively small size of the dataset, which may consequently limit the types of learning algorithms that could be applied to the data.

User Interaction User interaction is another aspect of this work. Controlling the synthesis process and editing the current result can be on semantic level as well as on the image level. Synthesizing images on the semantic level, such as specifying certain attributes like person's age, expression, moustache style, *etc.*, necessitates additional annotation of the data. Some aspects, such as illumination or camera angle, may be trained from synthetic data. On the other hand, adding constraints on the image level intuitively requires addressing the problem of human-computer interaction. For example, asking the user to specify RGB color constraints on any pixels is not easy to novices (*e.g.* choosing skin color is not trivial); or, drawing edges requires some artistic training, so some sort of visual feedback will prove helpful to the user. Moreover, editing the

structure map is another aspect of user interaction that has not been previously addressed. One of the challenges is providing "proxies" of image representation that are both intuitive to the user and tractable for the machine.

Modeling Appearance of Highly-Deformable, Non-structured Visual Objects Another challenge is a parametric model of the object appearance that can manage object pose variation explicitly. Overall, currently available part-based or fragment-based models (see section 2.1.3) only provide sparse set of pixels with descriptor information, which is not suitable for our purposes. An Active Appearance Model [8] does model some deformations of the objects, but allows only piecewise affine deformations as the images are subdivided into mesh grid. Layered Active Appearance Model [9] considers data as layers and can model missing segments and spatially separate labelings. However, this still requires matching of segments between images. Furthermore, to our knowledge there are no models that can address the structure variation.

Detail Hallucination It is expected that the statistical model would approximate images in the dataset either for tractability, for robust parameter estimation or to avoid overfitting. So, synthesizing high-quality images may require hallucination of high-frequency details on top of the blurry samples generated by the model. This is done by finding patch correspondences between the sample of the parametric model and the library of images. Naturally, such operations would need fast patch search and state-of-the-art blending of the patches (see section 2.2.1). Fast patch search is a challenging problem as datasets consist of hundreds of images each of which contains thousands or millions (taking into account multiple scales and rotations) of candidate patches.

Evaluation Lastly, the evaluation of the synthesized images remains a challenge. The realism and quality of an image is inherently perceptual. So, directly measuring the perceptual quality for a quantitative evaluation of the models would require user surveys. For example, the survey participants could be shown one image that is either a sample of the model or a real image of the library, and queried if the image is real or fake. An ideal model would be able to synthesize images that would trick a human to find them realistic. This test is a visual analogy to Turing's test [10]. Alternatively, the participants could be shown two images: a synthesized image and an image from the library, and asked which one is "more realistic". Unfortunately, in both scenarios the comparison of different models is sample dependant, both in the sampling of the training set and the sampling of the model for evaluation. For evaluation,

one needs to sample the model such that the samples cover the full space of visual object appearances. Hence, the number of samples would be at least in the hundreds, which limits the feasibility of such evaluations. There are quantitative metrics of image quality that measure the quality of an image given a "ground truth" reference image (*e.g.* structural similarity index measure [11]). These metrics could be used to measure how well does the model capture the appearance of the visual object by measuring the quality of the reconstruction of the image from the test set. Unfortunately, novel images that were randomly synthesized with the model would not have such a reference image for comparison. It is possible to approximate the quality of randomly generated images by testing the quality of reconstructions of the test set, which requires many more examples not used during training.

1.3 Contributions

First, we address the problem of synthesizing objects that have articulated pose variation. We focus on animals that may have self-occlusions. We propose a method for *interactive image synthesis*, which provides visual feedback to the user's drawing. The method computes a low-dimensional representation of the shape distribution in the training data from the parts annotations. This model can generate plausible shape suggestions to the user using an existing incomplete sketch as a query. We use the user's drawing of the shape of the animal and the specifications of the appearance to select a few exemplar images that are close to the input both in terms of shape and appearance. For image synthesis, we use a non-parametric approach, which generates hybrids of the selected unaligned exemplar images by exploiting part annotations. This data-driven method was evaluated with a user study. We compiled four datasets with hand-labeled segmentation masks and part labels. This method is described in Chapter 3.

Next, we addressed the problem of *capturing the appearance of highly-deformable, non-structured visual objects* by proposing Context-Conditioned Component Analysis (C-CCA) model. The C-CCA is a probabilistic generative parametric model, which is a generalization of Probabilistic Principal Component Analysis [5] and Active Appearance Model [8]. C-CCA represents data as a linear combinations of functions, rather than components. This allows conditioning the component functions on any complementary label data, for example, shape information such as semantic part labels. We derived an efficient training algorithm for the model and tested it on four datasets of varying

complexity. The model outperforms PPCA on the image reconstruction task. It successfully captures the appearance of the highly-deformable visual objects that have varying structure. We also demonstrated the model on appearance transfer and structured image inpainting tasks and obtained good results. The model is described in Chapter 4.

Finally, we investigated combining the parametric method and non-parametric method for image synthesis. We use C-CCA to build a low-dimensional parametric model of the data. Since the samples of C-CCA lack high-frequency details, we *hallucinate high-frequency details* by using patches of the training data. We can find correspondences between the patches of the C-CCA sample and the patches from the library, and stitch together the library patches to synthesize a high-quality output. However, the target visual objects are highly-deformable, so the patches extracted from the library can have arbitrary rotations and small scale changes, which significantly increases the complexity of the patch correspondences problem. To this end, we extend state-of-the-art patch correspondence algorithms [12] to exploit semantic labels in the training data and precompute correspondences between patches of different images in the training data. These correspondences can speed-up the candidate patch search in finding the correspondences between the output of the C-CCA model and the patches in the library. This approach is described in Chapter 5. While C-CCA requires semantic part labels map to draw a sample, designing a generative model of the shape of the visual objects is not in the scope of this thesis. We assume that the semantic part labels mask is either created with the help of the user, or it is generated by some shape model.

1.4 Publications

The interactive system for image synthesis described in Chapter 3 was published in the "Computer Graphics Forum" journal in 2015 [13]. The article was prepared in collaboration with Neill D.F. Campbell, Dan B Goldman and Jan Kautz. Specifically, Neill D.F. Campbell implemented the joint manifold discussed in section 3.4.2.

The Context-Conditioned Component Analysis model described in Chapter 4 was presented at the "International Conference on Computer Vision and Pattern Recognition 2015" [14]. The published article was prepared in collaboration with Neill D.F. Campbell, Simon J.D. Prince and Jan Kautz. Chapter 4 extends the published article with further quantitative evaluations in section 4.6.2 and discusses the relation of C-CCA to other models in section 4.7.

We intend to submit the combined methods of synthesizing images described in Chapter 5 to the "Eurographics" conference.

1.5 Outline

We first review relevant literature on image synthesis with parameteric, non-parametric and combined approaches in Chapter 2.

In Chapter 3 we describe our interactive sketch-driven image synthesis method. We synthesize images with a coarse-to-fine, locally-coherent, non-parametric algorithm which was inspired by state-of-the-art techniques in image inpainting and image hybrids.

In Chapter 4 we describe our globally-coherent parametric statisticial model called Context-Conditioned Component Analysis that captures the appearance of highly-deformable visual objects that have varying structure.

In Chapter 5 we describe how we synthesize images by sampling C-CCA and hallucinating high-frequency details on top of the output of C-CCA. Our approach finds patches from the training set that have similar appearance to the sample of C-CCA, but contain high-frequency details and then stitches them together.

Finally, we summarize the contributions of the thesis and outline future work in Chapter 6.

Chapter 2

Literature Review

The aim of this chapter is to give an overview of the current related work in image synthesis methods. As mentioned in Chapter 1, we categorize relevant literature into two approaches: sampling of parametric models and image synthesis with non-parametric models. Works that have elements of both the parametric and non-parametric models are discussed in section 2.3.

2.1 Parametric Models

We will first review models that exploit the global covariance of the pixels of the visual object and build a low-dimensional parametric representation of natural images. This requires answering at least two questions: (i) how to model color covariation of different pixels in an image; (ii) how to find the correspondence between pixels that should co-vary.

The discussion of related work in this section starts with Subspace Models that are only concerned with the first question by assuming that objects of interest in each image are spatially aligned to each other. This assumption is usually used for problems that model human faces. In such problems the images of faces are pre-aligned such that fiducial points such as corners of eyes have corresponding pixel coordinates in all images of the dataset.

The Deformable Models (section 2.1.2) and Part-based Models (section 2.1.3) propose two different approaches to the second question. Deformable Models assume that *all* pixels in each image must have some relationship to each other. Part-based Models assume that some pixels and their neighborhoods are more important than others, so the models should only be concerned with modeling important pixels and their relationships. Unfortunately, most of these models were built for object detection or object recognition problems; hence, they describe a sparse set of interest points or edges and the spatial relationships

between them. Consequently, most of the models are discriminative and not suitable for synthesis purposes. Nevertheless, some insights used in developing such models are applicable for image synthesis.

Next, we mention computer graphics approaches for synthesizing 3D shape models and plant modeling from images.

The last subsection discusses Neural Network approaches which have recently become very successful in image classification tasks [15]. Following the popularity of Neural Networks for discriminative tasks, there are recent works that use Neural Networks to build generative models that can synthesize images. However, the number of parameters in Neural Networks requires the training datasets to contain tens of thousands of images to avoid overfitting. Furthermore, increasing the resolution of images increases the number of parameters in the Neural Networks, so, typical generative Neural Networks synthesize images of low resolution (32x32 or 64x64).

2.1.1 Subspace Models

Subspace models define a probabilistic model that describes the modes of variation of all of the pixels in the image. For linear models such as PCA where the images are vectorized, the images in the dataset must have the same structure and have a good alignment. Thus, articulated objects may not be captured by linear models that are based on pixel intensities. However, if the object has the same structure and was captured in the same pose, such as a frontal image of a face with a neutral expression, the linear models can learn a good understanding of the global structure of the image.

Assume that we have N training images that were vectorized

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \tag{2.1}$$

The linear model assumes that the data can be represented as

$$\mathbf{x}_i = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i \tag{2.2}$$

where each $\boldsymbol{\mu}$ is the mean of the data, $\mathbf{F}$ is a factor matrix consisting of columns as the factors, $\mathbf{h}_i$ are the factor loadings or mixture weights. Turk and Pentland proposed a method called Eigenfaces [16] for face detection and identification, where the factor matrix consist of principal components derived from the covariance matrix of the probability distribution over the vector space of the face image pixel intensities. Similarly, one can model the shape

or some point distribution of a visual object by stacking feature points into a vector instead of pixel intensities. This approach has been introduced by Cootes *et al.* for locating visual objects, and is known as Active Shape Models (ASM) [17, 18, 19, 20]. For the model to capture the variations of the point distribution, the keypoints of each image have to correspond to each other. This is related to the part-based models since only a sparse set of keypoints is considered, however, ASM captures the geometric relation of all pairs of points (see Figure 2.1). Another shape model called Contour People was recently proposed by Freifeld *et al.* [21]. Contour People is learned from a dataset of 2D contours of different people in various poses that was generated from a parametric 3D SCAPE model [22] of the human body, which was learned from a database of laser scanned human bodies in various poses. Since the 3D data has body part segmentations, the generated 2D contours have ground truth body part annotations. The model decomposes the deformations of the contour into a product of three components: the shape variation, the part rotation and the viewpoint change. All these components were trained independently, since it is possible to fix each factor in the 3D model during the training data synthesis. The model was tested on the problem of human body segmentation and yielded good results. The key factor that enabled this model is the 3D model of the human body with ground truth part labels. Unfortunately, the dataset is not publicly accessible as it remains proprietary.

If we assume that the data is corruped with noise, each data point can be described as

$$\mathbf{x}_i = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \epsilon_i \ . \tag{2.3}$$

When the noise is assumed Gaussian with zero mean and diagonal covariance Σ, and we have Gaussian prior on the mixture weights, the model can be treated as fully probabilistic, where $\mathbf{h}_i$ are latent variables. The factors and the factor loadings can be learned using the Expectation Maximization algorithm [23]. Please see [24] for a unifying review of Linear Gaussian Models.

One has to choose the number of factors, which depends on the model and application (*e.g.* for images of faces: 8 factors in Visio-lization [1], 7 in Eigenfaces [16], 80 in Active Appearance Models [8], 16 in Tied Factor Analysis [25], *etc.*), so that the underlying subspace can be extrapolated by varying the factor loadings. Furthermore, the number of parameters that need estimating is directly related to the number of training samples. The reduced number of dimensions yields factors that are weighted averages of the

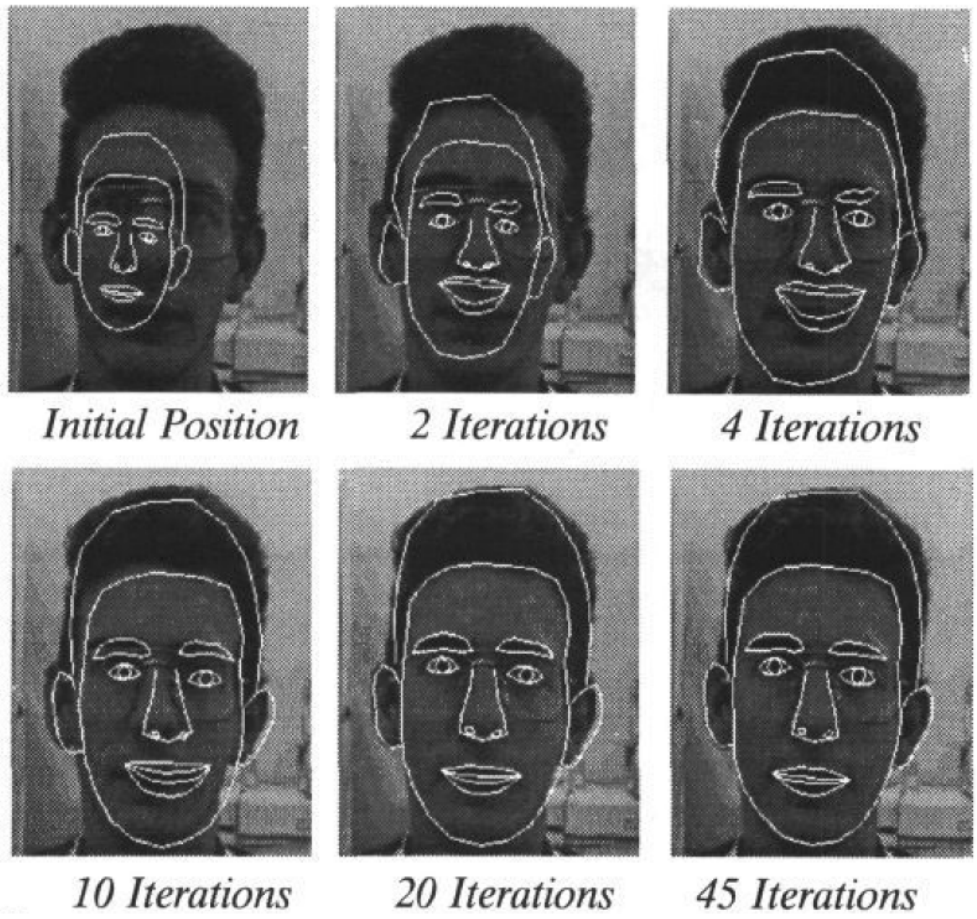


Figure 2.1: Example of a face ASM iterating to fit a new image. [Source: [17]].

datapoints. Hence, the reconstructions of datapoints are weighted averages of factors, which are weighted averages of datapoints in the training set. As a result, the datapoints are reconstructed as weighted averages of the datapoints in the training set. This averaging can be viewed as a low-pass filter applied to the datapoints, which attenuates high-frequency signals. Thus, the synthesized images lack detail as show in Figure 2.2.

Many related approaches followed; for example, the Fisherfaces algorithm [26] makes use of linear discriminant analysis (LDA). In this work, the subspace aims to maximize the ratio of inter-individual to intra-individual variance. The null-space LDA approach [27] makes use of the directions without intra-individual variance (these are ignored by Fisherfaces). The Dual-Space LDA approach [28] combined both of these subspace directions.

Recently, Prince *et al.* have proposed probabilistic latent discriminant analysis (PLDA) [29] to model intra-individual and between-individual variation

Figure 2.2: Random samples of the factor analysis model trained on images of faces. [Source: [1]].

of frontal face images for the face recognition problem. This was done in probabilistic manner, where the training set consisted of images of I individuals under J lighting conditions

$$\{\mathbf{x}_{i,j}\} \tag{2.4}$$

The model assumes that each image $\mathbf{x}_{i,j}$ can be expressed as a combination of the between-class factors and within-class factors. So,

$$\mathbf{x}_{i,j} = \boldsymbol{\mu} + \mathbf{F}\mathbf{h}_i + \mathbf{G}\mathbf{w}_{i,j} + \epsilon_{i,j} \ , \tag{2.5}$$

where $\mathbf{h}_i$ corresponds to the latent individual variable, which does not depend on the within-class factors, whereas $\mathbf{w}_{i,j}$ is the mixture weight that describes the within-class factors needed to explain the image $\mathbf{x}_{i,j}$. Lastly, the noise $\epsilon_{i,j}$ is modelled for each image separately. Under the assumption that $\mathbf{h}_i$, $\mathbf{w}_{i,j}$ have a prior of a Gaussian distribution with zero mean and identity covariance and $\epsilon_{i,j}$ is modelled as a Gaussian with zero mean and diagonal covariance Σ, the model can be learned with the EM algorithm. The model produced the state-of-the-art results with the keypoint related descriptors, and showed experimentally a good separation of within-class and between-class factors for the pixel intensity values. However, the model still suffers from overly smooth results. The same model was rediscovered by Gong *et al.* [30] to separate identity and age appearance factors for the problem of face recognition.

In another work, Prince *et al.* introduced the tied factor analysis [25] for the face recognition across large pose differences. The training set consists of I individuals with J examples of K poses

$$\{\mathbf{x}_{i,j,k}\} \ . \tag{2.6}$$

The image of an individual i under the j illumination and the pose k is modelled as

$$\mathbf{x}_{i,j,k} = \boldsymbol{\mu}_k + \mathbf{F}_k \mathbf{h}_i + \epsilon_{i,j,k} \ . \tag{2.7}$$

Here, each pose has a separate set of factors $\mathbf{F}_k$, and a mean face $\boldsymbol{\mu}_k$. Similarly to PLDA, each individual is modelled with a same latent individual vector $\mathbf{h}_i$. Again, under the assumption that the noise is Gaussian with zero mean and diagonal covariance Σ_k, and latent variable is Gaussian with zero mean and identity covariance, the model can be trained using the EM algorithm. Since the latent variable $\mathbf{h}_i$ is tied to an individual, it is possible to synthesize images of the same person under different poses. However, the result is still quite blurry and lacks details.

Manifolds Natural images do not necessarily lie on a linear subspace, but often are distributed on a low-dimensional non-linear manifold. There is a large body of work on manifold learning [31, 32, 33, 34, 35, 36], however, most of them assume that either the data points in the local neighborhood of the high-dimensional space should be nearby in the low-dimensional manifold or reversely the data points in the local neighborhood of the low-dimensional space should be nearby in the high-dimensional space. Other methods, such as [37, 38, 39, 40], do not provide a mapping from the latent space to the original space. Details of these assumptions are important for the construction of the manifold, but they have the same implication for sampling of novel points from the learned manifold: new data points are represented as a weighted average of a set of the training data points.

Some recent works use more powerful dimensionality reduction techniques to model images, especially for the problem of face recognition and identification, such as [41, 42]. However, the images are typically modeled as a weighted average of images in some local neighborhood, which results in artifacts similar to linear models.

2.1.2 Deformable Models

In this subsection we discuss methods that use dense warpings of images. As it was mentioned in section 2.1, finding corresponding pixels in different images is important for building a good low-dimensional representation. In this section we review methods that recognize that the camera angle, the pose or 3D deformations of the visual object are important factors that control the image of the visual object, and thus must be explicitly modelled. More specifically, the methods in this section model deformations with a dense warping. Nearly

all dense warping techniques assume a spatially smooth warping of the images. This assumption is reasonable for pairs of images that have similar structure and appearance, however, it corrupts warps for visual objects that have significant appearance variation (*e.g.* cats), cropping, or significant deformations that cause self-occlusions (*e.g.* legs of animals).

Active Appearance Model Active Appearance Model (AAM) is an extension of Active Shape Model that linearly models both the shape and the appearance of the visual objects [8, 43, 44]. So, AAM assumes the image of an object and a set of keypoints similarly to ASM as an input. The image is subdivided into a triangle mesh, where the keypoints are the vertices of the triangles, using DeLaunay triangulation [45] or some modification of it (for example, adding a constraint of equilateral-like triangles in the final mesh). Then, the images are warped to a mean template, and the pixel values are resampled. Thus, AAM has a better one-to-one correspondence of the pixel values than models that use the image coordinates.

While AAM is linear in both the shape and the appearance, it is not linear in relation to the pixel coordinates. Consequently, fitting AAM to an image is a non-linear optimisation problem. There is a large body of work on algorithms that minimize the error between the input image and the approximation provided by AAM [44, 46, 47, 48].

By modeling both the appearance and the shape of the visual objects, AAM can track the articulated objects like talking faces, *etc*. However, the allowed deformations are piecewise affine, which prohibits self-occlusions or other significant deformations. Moreover, AAM still performs dimensionality reduction, and the synthesized images still lack some detail. Lastly, AAM assumes a constant structure of the visual object. To solve some of these problems, a Layered Active Appearance Model (LAAM) was proposed by Jones and Soatto [9]. In this approach, the image is subdivided into layers, where each layer is associated with a set of landmarks points, a texture image and a local coordinate system. Each layer can be missing in some of the images. By computing the warps, the normalizations and the weights for each of the layers, and the data can be learned using the weighted PCA algorithm. However, this approach still requires correspondence of the landmark points and consistency of the layers.

Similarly to AAM, an approach called Morphable Model has also been applied for 3D face synthesis by Blanz and Vetter [49]. Here, the dataset consists of 3D face models, which is difficult to obtain. While Morphable Model

can synthesize the same individual under different viewing angles with better quality, it still suffers from the similar drawbacks to AAM. Patel and Smith [50] revisit 3D Morphable Models to improve the efficiency and general accuracy of the previous models. An interesting application of Morphable Model was demonstrated by Sucontphunt *et al.* [51, 52] who developed a method that allows to create 3D models of faces from a user's sketch. The method relies on parametric models of the 3D geometry (Morphable Models) and texture map (weighted average of texture maps of the dataset) of a face. While this method is successful for synthesis of 3D geometry and texture map of a face, the quality relies on the accurate alignment of geometries and texture maps in the dataset using the keypoint annotations. Hence, the range of visual objects that can be modeled with this approach is limited by the Morphable Models, as building 3D Morphable Models of non-structured visua lobjects is difficult.

Alignment-based Models AAM solves the deformation problem by training with the fiducial keypoints which are usually annotated manually. The natural extention that has generated attention is automatically solving for the deformation from the training images, *i.e.* obtaining dense alignment of images in the dataset to some common template or between pairs of images. Instead of manually annotating all images, Walker *et al.* [53] proposed to use temporal consistency in image sequences to automatically build appearance models by tracking stable fiducial points. Furthermore, Ramnath *et al.* [54] showed that iteratively increasing the density of the mesh of AAM makes fitting the new instances more robust. Recently, Krüger *et al.* [55] proposed to model the correspondences between landmarks of different images as probability distributions. This allows learning of the correspondence model from the training set of images, *i.e.* the probability distribution of the model correspondences conditioned on the observations of the image is jointly estimated from the training data. The advantage of modeling correspondences with probability distributions is that the images of the training set can have missing/unobservable landmark points. Given the learned model and a new test image, the model can be used to optimise the model-to-observation correspondences. Krüger *et al.* demonstrate the model on segmentation and classification tasks. Unfortunately, the model is conditioned on the appearance and cannot be sampled to generate new appearances.

There are multiple techniques of constraining the deformation to be smooth, for example, one approach is modeling deformations of images using a low-dimensional deformation field. Or, one can define deformations as local non-

overlapping transformations on a mesh similarly to AAM.

There is a large body of work on the problem of mesh deformation. These methods are usually used for character animation, so the focus is on getting deformations of characters in images or 3D shapes to look plausible. One of the most popular mesh deformation methods [56] is achieved through optimization of the mesh coordinates such that each triangle's deformation is as-rigid-as-possible. This method was used by Hornung *et al.* [57] to generate novel poses of people in 2D images. However, the result image does not have new appearance, only the new pose or shape.

To align images to each other, one must apply deformations to all images. If the color variation between images is not significant or can be factored out by image processing (for example, medical scans of organs, images of frontal view faces without background, *etc.*), one can define a suitable global alignment error metric such as per-pixel entropy across images. So, it is possible to reduce this global alignment error metric with iterative refinement of individual mesh vertices [58, 59, 60, 61, 62, 63, 64, 65, 66]. Liu *et al.* proposed SIFT flow [67] that solves the alignment problem by using SIFT descriptors [68] instead of pixel intensities in addition to smoothness constraints similar to optical flow. The alignment procedure itself does not produce an appearance model, but it is important for good alignment of the data when building appearance models.

Shape matching is a 3D formulation of estimating dense correspondences between pairs of images. The scenario of global correspondence estimation with non-rigid deformations in a collection of 3D shapes is an active area of research (see [80] for overview). Recently, Ovsjanikov *et al.* [79] proposed a novel representation for maps between pairs of shapes, which instead of correspondings points on the two shapes, puts in correspondence real-valued functions defined on the two shapes. This allows the mapping between the shapes to be represented as linear transformations between the corresponding function spaces. By choosing a suitable basis for the function space on each shape (they use eigenfunctions of Laplace-Beltrami operator defined on the shape) they yield representation of the map that is multi-scale, compact and suitable for efficient global inference. They achieve state-of-the-art results on isometric shape matching benchmarks using a standard set of point descriptors as map constraints. Adapting such representation of a map may be beneficial to the problem of 2D image alignment.

For objects with varying texture one must jointly learn the deformations as well as build an appearance model. Transformed Component Analysis [69]

estimates a global transformation for each image in the dataset to bring the images into alignment as well as finding an appearance subspace. The set of transformations they consider are rigid body motions, which are not suited to highly-deformable objects with self-occlusions. Winn and Jojic introduced LOCUS [70], an unsupervised generative model that learns segmentations of visual objects. Most interestingly, they model visual objects with deformation fields and achieve dense registration between different images of the same class despite differences in appearance or pose.

Mobahi *et al.* [71] also learn a model of a visual object from a set of images. They assume that images are generated as a nested composition of color, appearance and shape transforms. By modeling each component as a low-dimensional subspace, they can learn a regularized solution of the compositional model from a set of images. Their shape (*i.e.* geometric) transform is jointly learned for all images, and outperforms SIFT flow and robust optical flow for any pair of images from the set.

Besides the appearance, the 3D consistency of the learned shape model can be used to align images and and regularize the learned 3D shapes. In particular, the 2D projections of the 3D shape has to match the observed silhouettes and the 2D keypoints. Cashman *et al.* [72] showed how to build 3D Morphable Models of not strongly articulated visual object from 2D images with some user annotation. Among the results, the output 3D Morphable Model has high quality geometry for dolphin and pigeon examples, but not as high quality for bears. Vicente *et al.* [73] demonstrated that using ground truth class labels, segmentations and keypoints of Pascal VOC [74] dataset it is possible to compute dense, per-object 3D reconstructions. The method first estimates the camera positions from the keypoints in each method using rigid structure from motion algorithm [75]. Then, for each image, the algorithm finds most likely "surrogate" silhouettes from the other views using silhouettes of other images in the training set. Once, a triplet of silhouettes is estimated, the dense 3D shape can be computed by finding the visual hull of the shape [76]. Later, Kar *et al.* [77] showed how to learn linear deformable 3D shape models from a collection of images. The method adapts non-rigid structure from motion algorithm by Torresani *et al.* [78], such that it account for silhouettes. The method alternates between estimating (i) the camera extrinsics and the 3D locations of the keypoints (estimated from 2D using non-rigid structure from motion); (ii) the basis of the 3D shape fitted to the silhouettes of the training set; and (ii) the coefficients of the estimated deformation basis for each of the

images. Once the 3D shape model is learned, it can be used to reconstruct a 3D shape and high-frequency depth map from a single image. Unfortunately, these 3D shape methods do not investigate the appearance space, thus, they can be sampled to generate novel poses of the visual objects, but not the appearance.

2.1.3 Part-based Models

The part-based models capture the global appearance of a visual object by reasoning about a collection of local templates that deform and articulate with respect to one another, however, most of the part-based models are used for object detection problems. There are two core aspects: the descriptor representation used for the part templates and the model for expressing the deformation constraints. If no geometry is encoded in the model, this becomes equivalent to the "bag-of-words" models [81].

2D Relationship The original idea was introduced by Fischler and Elschlager in pictorial structures [82], and since has evolved and extended for object detection [83, 84, 85, 86].

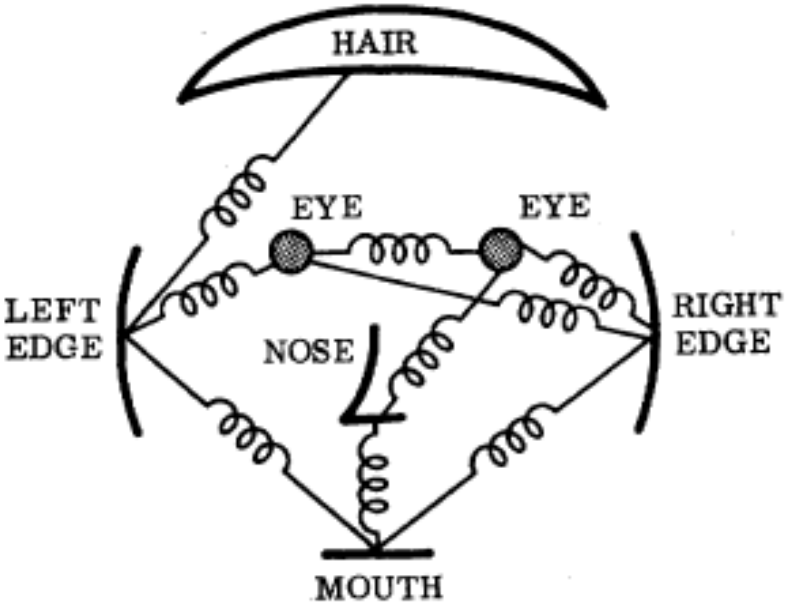


Figure 2.3: Illustration of the pictorial structures model of a human head. [Source: [82]].

A part could be described as a rigid template. Various descriptors are used for representing the rigid part templates, from directly modeling the pixel intensities to different histogram of gradients (HoG) features [87]. For object detection purposes, the introduction of certain invariances is generally

beneficial, because of the inherent complications such as cluttered background, occlusion, illumination, camera viewpoint, *etc*. Consequently, recent works tend to use HoG-like features that have some notion of the illumination and clutter invariance. Unfortunately, descriptors that ignore those appearance factors may not be suitable for image synthesis. Recently [88, 89], steerable part representations, which allow parts to be represented as a linear combination of a smaller set of basis parts, were shown to be fast to train, to require less memory and often reduce overfitting to the data.

A natural extension of Pictorial Structures model is allowing certain deformations of parts. Zuffi *et al*. [90] proposed such a Deformable Structures model, where each part is represented by a low-dimensional shape deformation space and the pairwise potentials between parts capture how the shape varies with the pose and the shape of the neighboring parts.

The major consideration for the geometric relationships of the part-based models is the availability of an efficient inference algorithm. Consequently, the part-based models have a sparse set of N parts and model the geometric relations between sparse pairs of the parts. One of the most popular forms is the so-called constellation model [91].

A particularly common constellation approach is a "star" model, which specifies the spatial location of the parts with respect to one root part, which is equivalent to modeling $N-1$ coordinates with respect to some common coordinate frame. Since only $N-1$ coordinates are kept, it is possible to have a lot of parts, generally called as vocabulary of visual words. For the face localization problem, Burl *et al*. [91] used a supervised (hand-picked) selection of interest points as parts. Leibe *et al*. [92] proposed a method that is similar to the "star" model, where the detected parts perform Hough voting on the object's center location for the object detection. Later, unsupervised ways of selecting the parts and extracting the geometric relations were proposed [93, 94, 95].

A generalization of the star model is a tree model, where each part's location is independently defined with respect to its parent part's location. While this approach can model more complex geometric relations, it still has efficient inference algorithms. The construction of the tree from the parts needs careful consideration, as multiple parts can independently fit to the same image region. One of the recent successful works was by Zhu *et al*. [96], where they presented a unified model for the face detection, the pose estimation and the landmark localization, while trained on hundreds of images, rather than thousands or billions. The model is a mixture of trees for different viewpoints of

the face. The trees share the pool of the parts; however, each viewpoint tree has a separate linear mixture vector that is used to define the tree's part templates as a linear combination of the templates in the pool of the parts. The parts were described using HoG features. Since the deformation model is captured by a tree, loops of landmarks, such as the landmarks around the mouth, are not explicitly constrained. It is still easier to optimize and it performs better than Active Appearance Model. However, discriminative part representation and the lack of the loopy constraint makes this approach unsuitable for the synthesis purposes.

Wu *et al.* [97, 98] proposed "Active Basis" as a way of representing a deformable template. So, the template is approximated with a linear combination of Gabor wavelet elements that are allowed to have spatial shifts and rotations. The elements are shared between all templates of each object class, but the deformations are individual for each image. Later, Si and Wu [99] extended Active Basis to a sparser representation called Shape Script. Although these models are generative, they were designed for effective object detection and recognition, not for synthesis.

2.5D Relationship Recently, an alternative representation of parts called "poselets" [100] was applied to the human detection, the segmentation and the body pose estimation problems. Unlike other part-based models, the poselets do not assume that parts have to correspond to the human anatomy, so the poselets have broader definition of the parts, which may correspond to the multiple parts or even the whole human body. The main motivation is that poselets should be easy to find in an image and they should be easy to localize the 3D configuration of the relevant keypoints, given the detected poselet. This provides an intermediate representation of the possible 3D locations of keypoints, which may be refined using the geometric constraints. A data-driven approach was introduced to estimate the poselets and train poselet detectors. The position of the keypoints are inferred by Hough voting [101] of the detected poselets. The poselets model was extended [102] to the structured hierarchical model, which is more sophisticated than simple Hough voting scheme. Pishchulin *et al.* [103] proposed to condition tree-structured pictorial structures model with higher-order geometric part dependencies of poselets for the human pose estimation problem.

3D Relationship Most of the part-based models represent the data in 2D by describing each part using 2D descriptors and modeling the geometry with 2D transformations. However, the data is generated from 3D objects, and

the geometric relations should be 3D transformations. A novel representation of the 3D object classes was proposed by Savarese *et al.* [104, 105, 106, 107]. The parts are modelled as large and discriminative regions consisting of local invariant features, for example, regions with bag-of-words as histograms. The geometric relations are modelled as homographic transformations between parts. This representation allows synthesizing novel views and unseen poses of the object classes. The synthesized views contain sparse set of features, which is sufficient for recognition, detection and categorizations tasks (Figure 2.4). Later, Su *et al.* [108] proposed a method based on this representation to learn a dense multi-view representation for detection, viewpoint classification and novel view synthesis. However, the dense representation is achieved by morphing a triangular mesh of view-points, which assumes consistent structure of the visual objects. Furthermore, the synthesized image has a novel viewpoint, but the visual object does not have a novel appearance.

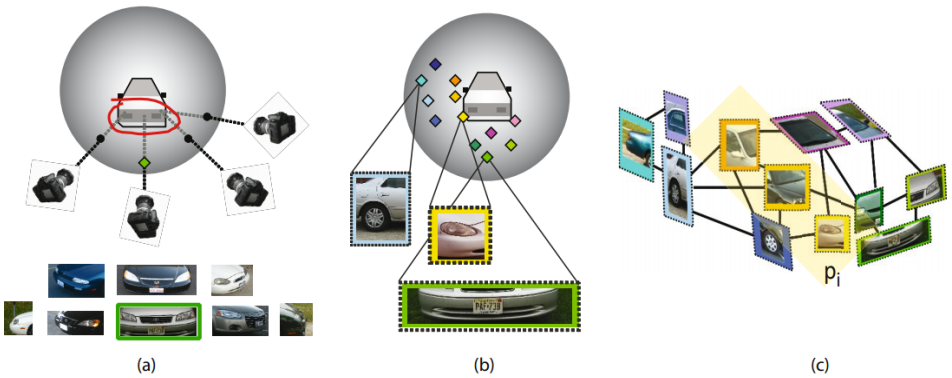


Figure 2.4: Summary of the Model of Savarese and Fei-Fei. (a): A car within the viewing sphere. As the observer moves on the viewing sphere the same part produces different appearances. The location on the viewing sphere, where the part is viewed the most frontally, gives rise to a canonical part. The appearance of such canonical part is highlighted in green. (b): Colored markers indicate locations of other canonical parts. (c): Canonical parts are connected together in a linkage structure. The linkage indicates the relative position and change of the pose of a canonical part given the other (if they are both visible at the same time). This change of location and pose is represented by a translation vector and a homographic transformation respectively. The homographic transformation between canonical parts is illustrated by showing that some canonical parts are slanted with respect to others. A collection of canonical parts that share the same view defines a canonical view (for instance, see the canonical parts enclosed in the area highlighted in yellow. [Source: [104]].

Mid-level Patches An important problem of part-based models is finding

the set of patches that correspond to a part from the training data in an unsupervised fashion. These patches have to be detectable (an algorithm should be able to detect it from the image), discriminative (*i.e.* they look differently from the patches of other objects) and representative (*i.e.* they occur frequently in images of objects). These parts are often called mid-level patches and there are a lot of methods that tackle this problem [109, 110, 111, 112]. These patches can be used to summarize architectural styles of cities [113], keep track of changes in style through time [114, 115] and help to align images in big image collections for exploration and visualization [116]. Moreover, such mid-level patches are useful for object detection problems in general. However, even if such patches are identified in a dataset, it is not clear how they can be used to synthesize novel images.

2.1.4 3D Shape Modeling

In this subsection we review methods that generate 3D shape models. We have already mentioned Morphable Models, which generates both the 3D shape model and the appearance (*i.e.* texturing) of the visual object.

A solution to the problem of synthesizing novel 3D shapes was proposed by Kalogerakis *et al.* [117]. The input data consisted of 3D shapes with consistently segmented components. The model learns structural variability by relating probabilistic relationships between geometric and semantic properties of shape components. Due to the consistent segmentation and labeling of components, complete 3D information and absence of sensor noise or occlusions, this model performs exceptionally well on 3D shape datasets. However, this approach is difficult to translate to 2D image synthesis.

Ovsjanikov *et al.* [118] built a navigation interface for exploration of a collection of 3D shapes. Their technique assumes that it is possible to build a low-dimensional manifold of 3D shapes in some descriptor space which does not require correspondences between shapes. However, they do not interpolate or synthesize novel shapes.

There is a large body of work on modeling and generating trees, either from video sequences [119] or images with user annotations [120, 121, 122]. However, they use domain knowledge specific to trees, *i.e.* branching, skeletal structure of the trees, *etc.* Quan *et al.* [123] designed an interactive system for image-based plant modeling that allows leaf and branch structure editing. The result is a textured 3D model which models the plant in the image rather than a generative model of a plant.

2.1.5 Parametric Texture Synthesis

In this subsection we review parametric models for the problem of texture synthesis, where a small image of a texture is used as an input to generate a larger image with the same texture. Early approaches [124] were focused on modeling the statistics of the target texture's pixel intensities. Rather than matching statistics of pixel intesities, some works [125, 126, 127] improved the quality of synthesized texture by optimizing the image with respect to statistics of linear filter responses computed for the target texture.

In both cases the general framework of texture synthesis is as follows. First, linear feature responses are computed for the target texture image. Next, statistics on the feature responses are computed over the target image in some spatial window. Finally, a new image is synthesized from a random white noise image by performing gradient descent with respect to the target statistics. These methods would perform well on the highly stochastic textures, but would fail to generalize to all natural textures, and also were outperformed by non-parametric texture synthesis approaches (see section 2.2.1) at the time.

Recently, Gatys *et al.* [128] used a similar framework for texture synthesis, but instead of matching the statistics of the linear filter responses, they match the statistics of non-linear responses of a Convolutional Neural Network (see section 2.1.6). This approach has dramatically improved the quality of synthesized textures.

Later, Gatys *et al.* [129] used different layers of the Convolutional Neural Network to separate the style and content of an image (this is similar to texture-by-numbers discussed in section 2.2.1). Two images are used as input to the system: style image (*e.g.* a painting of a famous artist) and content image (*e.g.* a photo taken by a user). The aim is to generate an image that has the global layout of the content image, but the local style of the style image. The method computes the statistics (Gram matrix of activations) of low-level layers of the Convolutional Neural Network of the style image and activations of the high-level layers of the content image. Then, the output image is optimized, such that the low-level layer statistics match the statistics of the style image, whilst the high-level layer activations match the content image. The motivation is to preserve the global arrangement of the content-level constraint and match the color and local statistics of the style image. Later, Li and Wand [130] improve content-by-example synthesis of Gatys *et al.* by replacing the Gram matrix term with a Markov Random Field term that minimizes the Euclidean

distance between the high-level activations of the output image and the style image. As both of these methods are input-driven, it is not clear how this approach can be generalized to generate various contents or styles from a library of examples.

2.1.6 Neural Networks

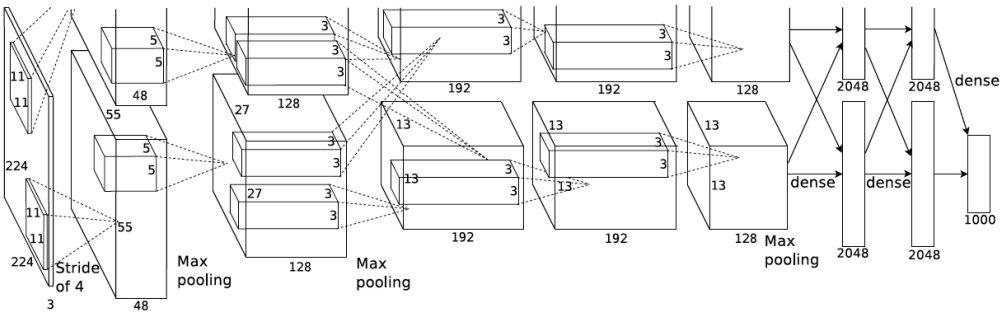


Figure 2.5: Architecture of the Deep Convolutional Neural Network of Krizhevsky *et al.* [Source: [15]].

In this subsection we discuss Neural Network approaches. Recently proposed, novel techniques for training and weight regularization allow to train deep architectures [131]. There are multiple architectures of Neural Networks that are designed for different tasks. Nearly all deep Neural Network architectures have the number of parameters in the millions, which results in the requirement of a vast amount of training data.

Encoder Neural Networks One of the recent successful works, Krizhevsky *et al.* [15] used Convolutional Neural Network architecture for a discriminative problem of object detection that achieved the state-of-the-art performance at the time. The CNNs architecture (see Figure 2.5) consist of multiple convolutional layers, where each layer convolves the input with a set of filters, passes the responses through non-linear function and max-pools multi-channel layers of representation. The first layer starts with the raw RGB pixel data. Max-pooling operation collapses a local region of the input into a single node, thereby reducing the size of the input to the next layer and increasing the "field of view" of the neurons in the next layer. The top layer (or top 2 layers) is usually "fully connected", meaning each neuron processes all of the nodes in the previous layer. In object recognition task, the output of the top layer is a vector of values with the size corresponding to the number of classes in the object detection problem. While Neural Networks outperform other machine

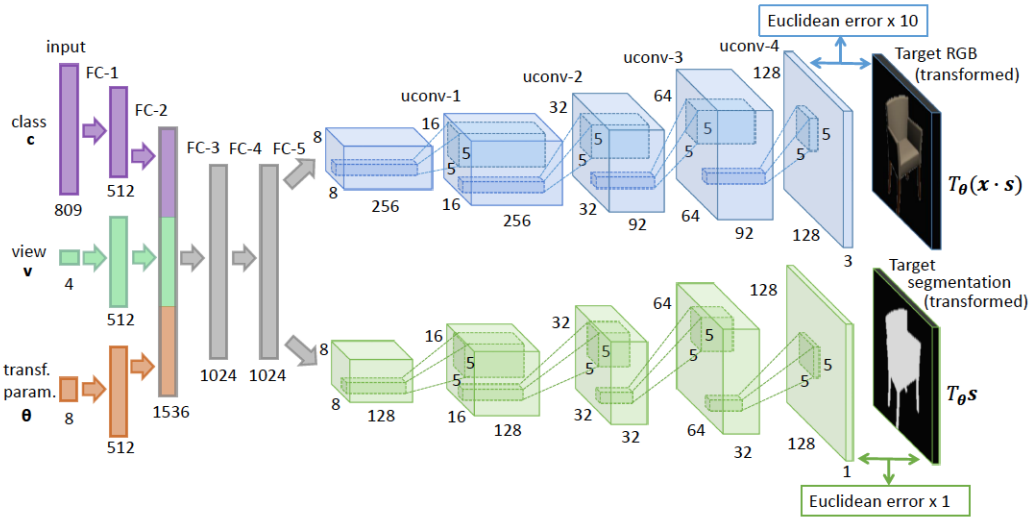


Figure 2.6: Architecture of the De-Convolutional Neural Network of Dosovitskiy *et al*. [Source: [132]].

learning techniques, the analysis of the trained networks proves to be difficult. An interesting approach was proposed by Mahendran and Vedaldi [133] for visualizing the behaviour of the Neural Networks. They propose to directly optimize the image that is input the trained Neural Network, such that the activations of the Neural Network match the target activations of the nodes in the network. The realism of the generated image is improved by constraining the target image with appropriate image priors. This approach is similar to parametric texture synthesis (see section 2.1.5).

Decoder Neural Networks The CNN architecture can be reversed top-to-bottom, where the goal is to "de-convolve" [134] a low-dimensional representation of an object to the full resolution. For example, Dosovitskiy *et al*. [132] trained Convolutional Neural Network to learn to generate images and segmentations of chairs from a dataset [135] of renderings of 3D models of chairs. Figure 2.6 visualizes the architecture proposed by Dosovitskiy *et al*. The CNN was trained using ground truth semantic parameters such as chair class, viewing angle, *etc*. One interpretation of this method is that CNN learned to "render" chairs from its semantic parameters, rather than CNN learned the appearance representation from the images of chairs. Nevertheless, random semantic parameters can generate images of size 128x128. The ability to interpolate between different viewing angles or chair styles allows this network to be used for the state-of-the-art correspondence estimation. The synthesized images have artifacts, especially images of chairs of interpolated chair styles, which restricts novel

image synthesis. Moreover, the dataset images were rendered under single illumination which consequently is not modelled by the CNN.

Encoder-Decoder Neural Networks There are multiple architectures that combine Encoder and Decoder networks. The goal is to have a model that computes low-dimensional latent representation by running the high-demensional input through Encoder network and ability to reconstruct the high-dimensional input by running the low-dimensional latent representation through Decoder network. This is similar to parametric models (section 2.1), but the Neural Network allows to learn the non-linear kernels by minimizing difference between original input to the encoder and the reconstructed output of the decoder.

One such model is called autoencoder. Hinton and Salakhutdinov [136] showed how to train autoencoders using the Restricted Boltzmann Machine [137, 138] with the encoder and decoder networks sharing weights for the problem of dimensionality reduction. The Deep Boltzmann Machine inference, or sampling, is not straightforward as the top-most hidden layer's values depend on the previous layer. Hence, Markov Chain Monte Carlo sampling techniques are required for generating posterior probability distribution.

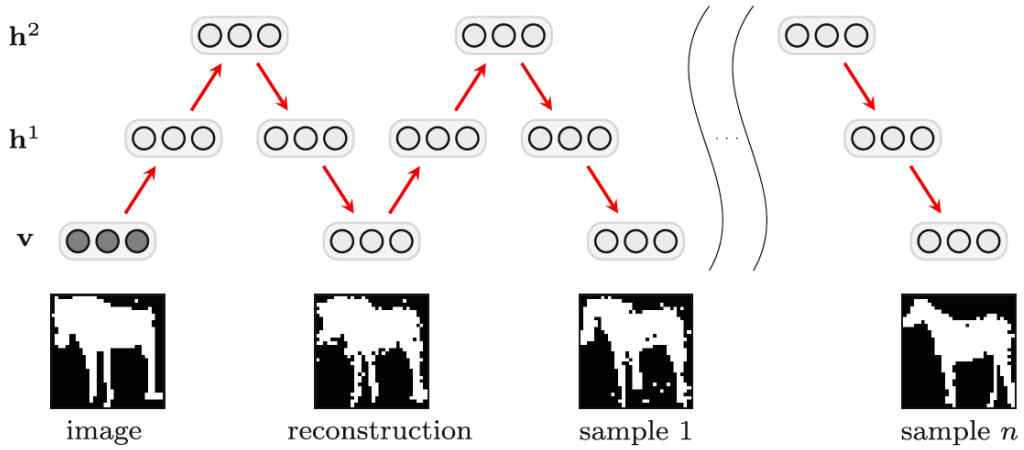


Figure 2.7: Sampling of the Shape Boltzmann Machine by Eslami *et al.* [Source: [139]].

Eslami *et al.* proposed Shape Boltzmann Machine [139] approach for modeling the object shapes. The work is based on the Restricted Boltzmann Machine and the Deep Boltzmann Machine [140] technique with the modification of sparse connections for the tractability. The task is to model a binary shape image. Figure 2.7 illustrates Block-Gibbs Markov Chain Monte Carlo sampling of the new shapes from the Shape Boltzmann Machine. The results clearly outperform alternative approaches for the subspace modeling, but still have

artifacts. Moreover, the binary images have size of only 32x32 pixels for the horses and other categories, and 64x64 for the motorbikes, which is far from desired. Authors designed the model to be used as a shape prior for the object segmentation tasks. This method was later extended from segmentation masks to part masks [141, 142].

Recently, Nhan Duong *et al.* [143] trained a Deep Boltzmann Machine using RGB images and fiducial keypoints similarly to Active Appearance Model (see 2.1.2). Quite expectedly they outperform Active Appearance Model on the task of face image reconstruction as their model is non-linear unlike AAM that uses PCA for dimensionality reduction.

Generative Adversarial Networks Goodfellow *et al.* [144] proposed Generative Adversarial Networks architecture that also consists of two networks: the generative network and discriminative network. The generative network's architecture is similar to decoder, and the discriminative network's architecture is similar to encoder. The generative network's objective is to learn to generate samples that are similar to the training set from a random input code. The discriminative network's objective is to learn to differentiate between images of the training dataset and synthesized images of the generative network. The Generative Adversarial Networks are effectively playing a minimax two-player game. Radford *et al.* [145] investigate constraints on the architectural topology of Convolutional Generative Adversarial Networks for robust and stable training. Denton *et al.* [146] trained a Laplacian pyramid of Generative Adversarial Networks. The approach uses Laplacian pyramid where the first layer is a noise sample and each layer upsamples the previous layer and adds correction to the layer computed by the generative network. This procedure effectively creates a super-resolution of the initial noise sample.

The adversarial framework of unsupervised representation learning not only is difficult to train, but requires a lot of data due to the number of parameters in the model. The methods train on datasets of rather small images, specifically 32x32 for CIFAR-10 dataset [147] and 64x64 for LSUN dataset [148].

Recurrent Network The DRAW (Deep Recurrent Attentive Writer) model of Gregor *et al.* [149] use a Recurrent Neural Network and an attention mechanism to generate images. Specifically, the spatial attention mechanism defines a trajectory of a "brush" which sequentially applies patterns on top of a canvas to generate an image. In effect, the model uses autoencoder network that uses Recurrent Neural Network both for the encoder and decoder. Hence, the latent representation of the autoencoder is in fact a sequence latent values. The model

was evaluated on the datasets of images of size at most 32x32.

2.2 Non-parametric Models

In this section we will review works that consider non-parametric image synthesis. The focus of such approaches tends to be on the local coherence of the pixel intensities without explicitly modeling the global coherence. The most basic approaches address the problem of texture synthesis, where a small image of a texture is used as an input to generate a larger image with the same texture. Unlike parametric texture synthesis 2.1.5, this section discusses works that treat the problem of texture synthesis as a non-parametrical distribution sampling. This approach was also used in such applications as image inpainting, painting-with-numbers, image morphing, etc.

2.2.1 Non-parametric texture synthesis

The non-parametric texture synthesis was introduced by Efros and Leung [150] and was inspired by the n-grams method for generating English-like text proposed by Claude Shannon [151]. The principle was to model the texture as a Markov Random Field, where each pixel has a square window around the pixel as its neighborhood. Then, for synthesizing a new pixel, one has to find a pixel in the source image that has similar neighborhood to the target pixel. This method is somewhat sensitive to the window size and may suffer in the regions where the intensities are uniform.

Texture synthesis methods can be used for the image inpainting problems. Here, the aim is to estimate the pixel intensities in some region such that the result is coherent with the rest of the image. The basic principle is to fill in the holes inside the missing region using the boundary pixel values as the constraint. This approach requires an efficient algorithm for finding the patches that have good correspondence in the overlapping area.

Wei and Levoy [152] proposed an accelerated algorithm for texture synthesis that uses multi-resolution synthesis and tree-structured vector quantization for fast nearnest neighbor matching of patches. Ashikhmin [153] built on the Wei and Levoy's algorithm and proposed to exploit spatial relation between matching pixels and copy chunks of pixels once a suitable pixel match was found. This idea was later developed by Barnes *et al.* [154] (see below 2.2.1).

Efros and Freeman proposed "Image Quilting" [155] method that stitches together small patches of the source image in a consistent way, which resulted in a more robust and faster method for the texture synthesis and transfer

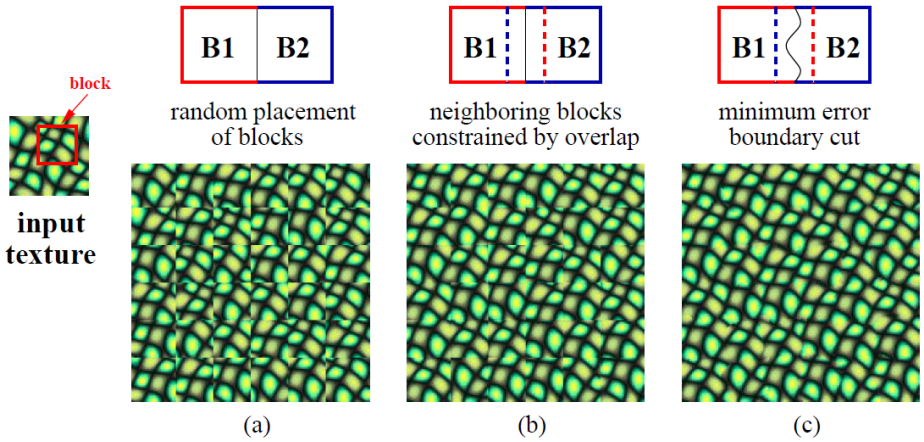


Figure 2.8: Quilting texture. Square blocks from the input texture are patched together to synthesize a new texture sample: (a) blocks are chosen randomly, (b) the blocks overlap and each new block is chosen so as to "agree" with its neighbors in the region of the overlap, (c) to reduce the blockiness, the boundary between blocks is computed as a minimum cost path through the error surface at the overlap. [Source: [155]].

problem. The method starts from the upper left corner and goes through the target image from left to right in a raster scan order. The first block is chosen randomly. The next step is to find a patch from the source image that minimizes the overlap between the previous patches to the left and above. Once such a patch is found, a minimal cost path through the overlapping area is computed using dynamic programming. The new patch is copied with respect to the computed path. Figure 2.8 illustrates the result. Unfortunately, this approach lacks notion of global coherence, thus, it is not applicable for images where different regions have different texture, such as faces, buildings, animals, *etc.*, see Figure 2.9 for an example.

Unlike image quilting that has a predefined window size, a more general approach was introduced by Kwatra *et al.* [156], where a graphcut technique [157] is used to compute seams of the patch regions from the source image and optimize for the new patch location. This framework also enables easier extension to the higher dimensions and the addition of other constraints.

Recently, the PatchMatch [154] algorithm was proposed to solve this correspondence problem efficiently. Each pixel in the missing region has a corresponding index to some patch in the rest of the image. This index map is initialized at random. First, the algorithm computes the score of how good is the match by comparing the error across a small patch centered on the pixel

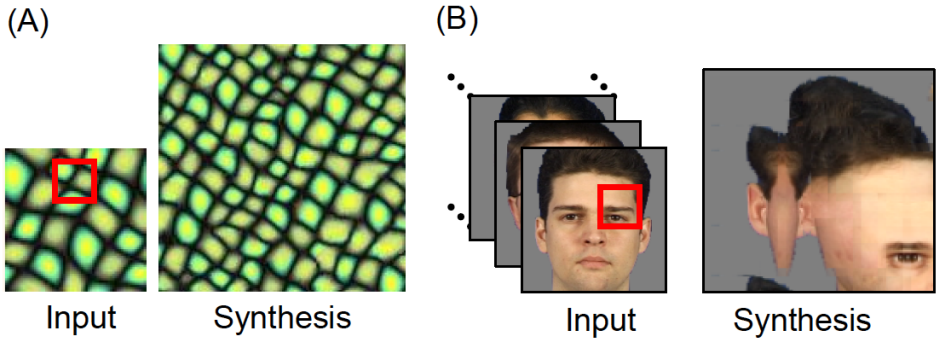


Figure 2.9: (A) Texture synthesis from input exemplar. (B) Generating faces with the texture synthesis algorithm. The synthesized image lacks global coherence. [Source: [1]].

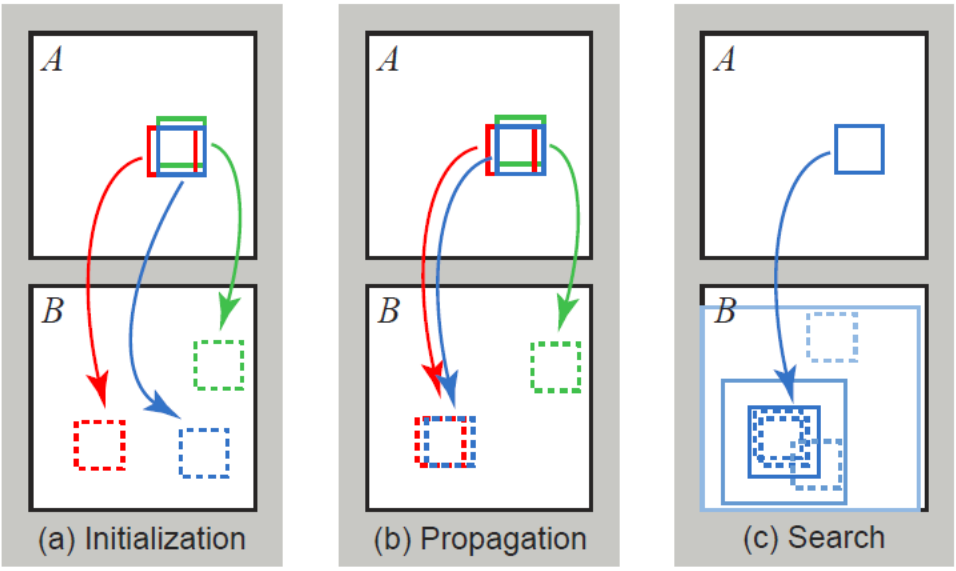


Figure 2.10: Phases of the randomized nearest neighbor algorithm of Barnes *et al.*: (a) patches initially have random assignments; (b) the blue patch checks above/green and left/red neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods. [Source: [154]].

and the corresponding patch from the rest of the image. The next step is the propagation, which refines the index map by looking at the neighbors of the good matches in the missing region and the neighboring patches in the indexed patches. The intuition is that the good matches are likely to have the spatial neighbors that would also be good matches if indexed in the same region. Finally, the patches are searched randomly for better matches. Figure 2.10 illustrates these stages.

The PatchMatch was extended to consider rotations and mirroring the patches in the Generalized PatchMatch algorithm [158]. A novel approach in combining different textures was introduced in the Image Melding method by Darabi *et al.* [159]. Multiple works have developed approximate nearest neighbor patch search methods to cases of large image datasets [160, 161].

In virtual reality and computer graphics applications there is a problem of texturing the virtual world. This has resulted in adapting texture synthesis algorithms to be able to synthesize facades and architectural objects [162, 163, 164, 165]. One of the problems is that the source images have multiple independent architectural elements such as doors and windows which need to be reproduced in the synthesized result without stretches and croppings. Typically, this requires horizontal and vertical coherence of elements. So, extracting valid structure maps from images [166] or editing elements of the facades in coherence with the structure [167] are important subproblems of building facade texture synthesis approaches.

Ying *et al.* [168] showed how to synthesize texture on 3D surfaces, rather than 2D planes, thus, dealing with artifacts of conventional texture mapping. Fang and Hart [169] developed an editing tool called "Textureshop" that allows clever texture transfer with realistic 3D deformations. First, the system estimates 3D normals of the target image using shape-from-shading technique. Next, it clusters the normals to create superpixels. The texture synthesis method is applied to the superpixels with the constraint of consistent neighborhoods. The final image is synthesized by applying patch distortions such as patch orientation in 3D, displacement mapping from estimated local surfaces and edge-like feature matching constraint of the texture synthesis.

Enriching Appearance Space One of the artifacts of the patch-based texture synthesis is the inconsistency on the boundaries and edges which is caused by insufficient number of matching patches in the source image. Wu and Yu [170] proposed to use a precomputed feature map as an additional channel that constrains texture synthesis process. The feature map encodes curvilinear features and their deformations.

Another application that uses similar principals is so-called "Image analogies" proposed by Hertzmann *et al.* [171] Here, the input to the algorithm consists of 3 images: A, A' and B, where image A' is a "filtered" or "modified" version of A. The goal is to synthesize image B' that has the same relationship to B as A' has to A. Figure 2.11 illustrates the problem. The algorithm performs a lookup of a patch in the image B that is the closest in some distance

Figure 2.11: Example of image analogy of Hertzmann *et al*. The problem is to compute a new "analogous" image B' that relates to B in "the same way" as A relates to A'. Here, A, A', and B are the inputs to the algorithm, and B' is the output. [Source: [171]].

space to some patch in the image A. Given the index, the algorithm retrieves corresponding pixel from the image A' into the image B'. This formulation of the problem allows various applications such as learning the image filters, super-resolution, texture transfer and texture-by-numbers. Most interesting for this literature review, is the texture-by-numbers application, best described by the illustration in Figure 2.12. For texture-by-numbers, the input image A is made of the sparse colors that describe the general layout of the image. This enables modification of the layout in the image B. The algorithm produces the image B' that is textured similarly to the image A', but with the layout guided by the image B.

Diamanti *et al*. [172] proposed a system that combines texture-by-numbers and image melding approaches. The system provides high-level image editing, such as retexturing objects by using user annotations.

Recently, a GPU powered approach was proposed [173] to solve the texture synthesis problem, where the texture is synthesized pixel by pixel. The approach synthesizes texture in the coarse-to-fine fashion and works with the index maps as proxies rather than directly with the pixel values. At each level, the index map is upsampled from the previous level. Then, the index map is jittered, to add randomness to the result. Finally, the jittered result is corrected, so that the pixel's neighborhoods in the output image are similar to those in the source image. All these stages can be implemented on the GPU for efficiency, which is desirable for the graphics applications. Figure 2.13 illustrates the stages of the algorithm. Later, this approach was refined [174] by encoding the pixel values in the non-local appearance space, rather than RGB space. This is motivated by the patch-based texture synthesis methods. In the source image, each pixel is encoded using RGB values. Then, a 5x5 patch around each pixel is encoded into a new appearance-space E. This image E has 5x5x3 dimensions. For

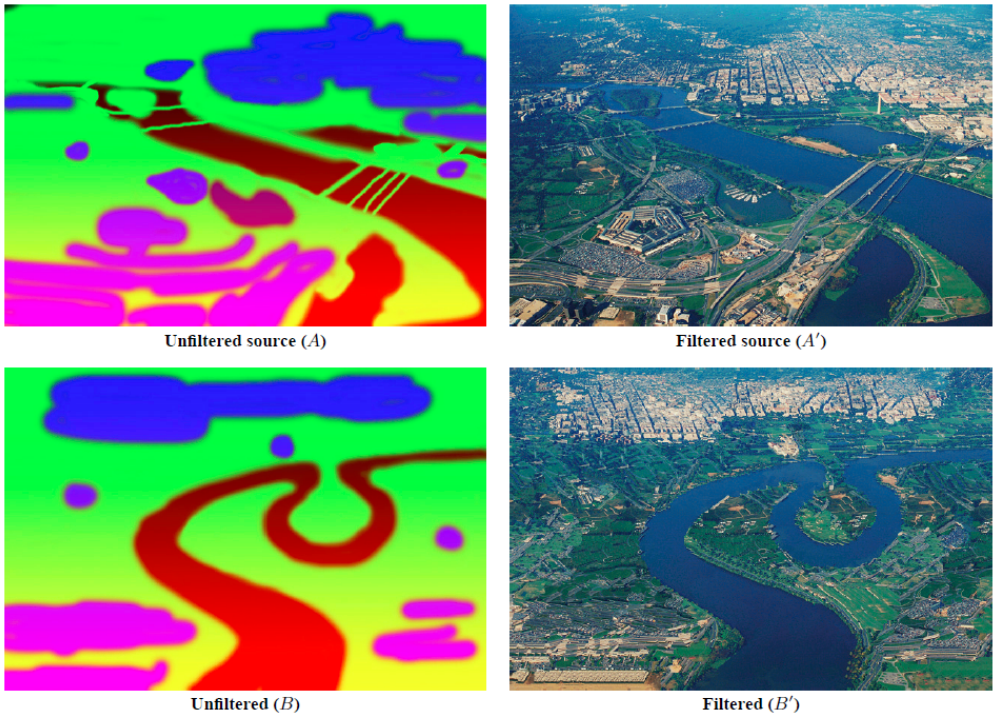


Figure 2.12: Example of texture by numbers of Hertzmann *et al*. The unfiltered source image A was painted by hand to annotate A'. The unfiltered target image B was created in a paint program and refined with an interactive editor; the result is shown in B'. Ordinary texture synthesis cannot reproduce the terrain in the photograph because it is not stationary: far elements are different from near elements. The use of the gradient channel in A and B distinguishes near from far, allowing the photograph to be used for texture-by-numbers. [Source: [171]].

efficiency and desirable irregularity of the result, the 5x5x3 dimensional space is projected on 8 dimensional appearance space E' using PCA. This new space is used for the pixel-based texture synthesis. As a result, larger neighborhood information is indirectly encoded into the local overlapping regions.

Using Aligned Exemplars The appearance space and the three-stage texture synthesis algorithm were reformulated by Risser *et al*. [4] for synthesizing Structured Image Hybrids. Rather than generating a larger image with the same texture, the image hybrids approach retains the structure of the image while introducing diversity by mixing together multiple aligned exemplars of the same visual class. The method works in a coarse-to-fine fashion, with similar stages to the texture synthesis. The main difference is in the jitter stage, where instead of introducing x, y coordinate jitter, the method jitters between the different exemplars as shown in Figure 2.14. This was successfully applied

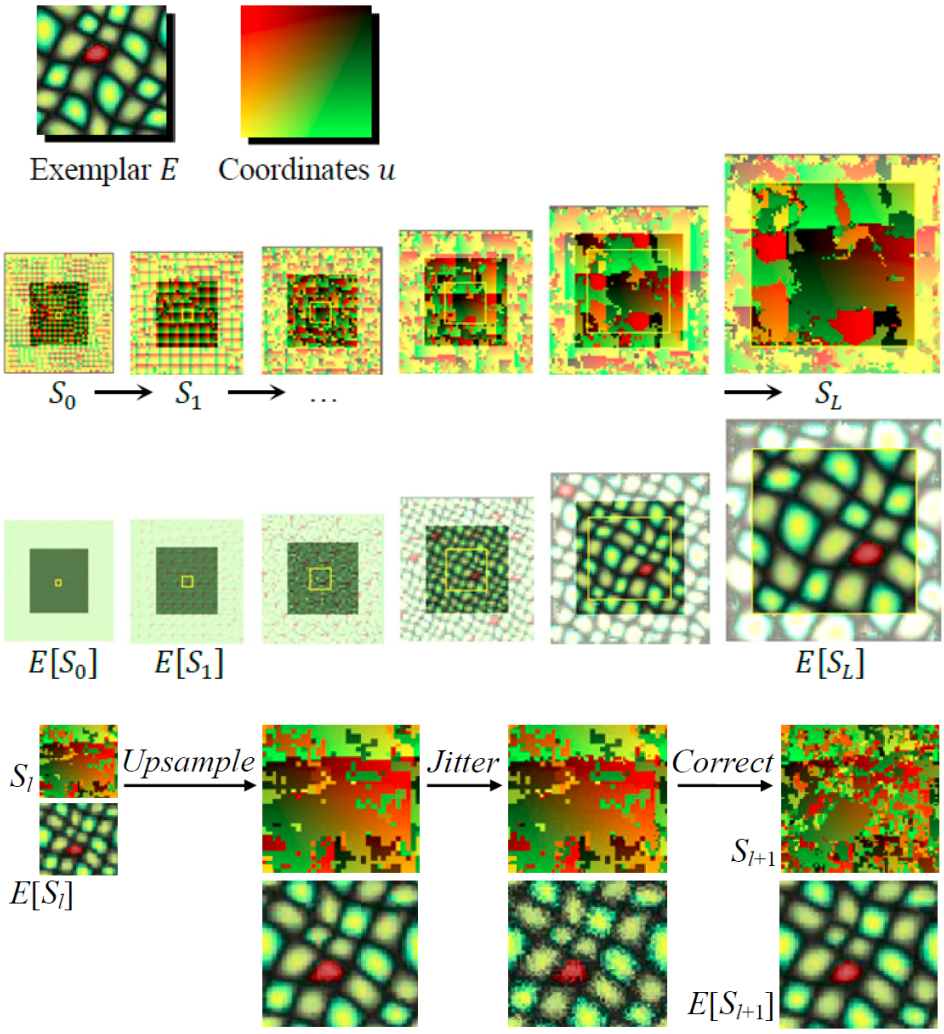


Figure 2.13: Parallel texture synthesis framework of Lefebvre and Hoppe with the three stages of the texture synthesis at each pyramid level. [Source: [173]].

on different examples such as jeans, faces, butterflies, *etc*. See Figure 2.15 for examples. Notice that multi-scale approaches do have some notion of non-local coherence, but do not explicitly model the global coherence.

In a similar fashion, Tappen and Liu [175] use SIFT flow to align images of faces from a database to a target low-resolution image and use the aligned images as source exemplars to perform hallucination of high-resolution image details. Although the generated image has novel details, this approach requires a valid, globally-coherent, low-resolution image as input.

These non-parametric models improve image synthesis through alignment, however the global coherence of the images is preserved either through the alignment of the exemplars (Risser *et al*. [4]) or by smoothly warping images

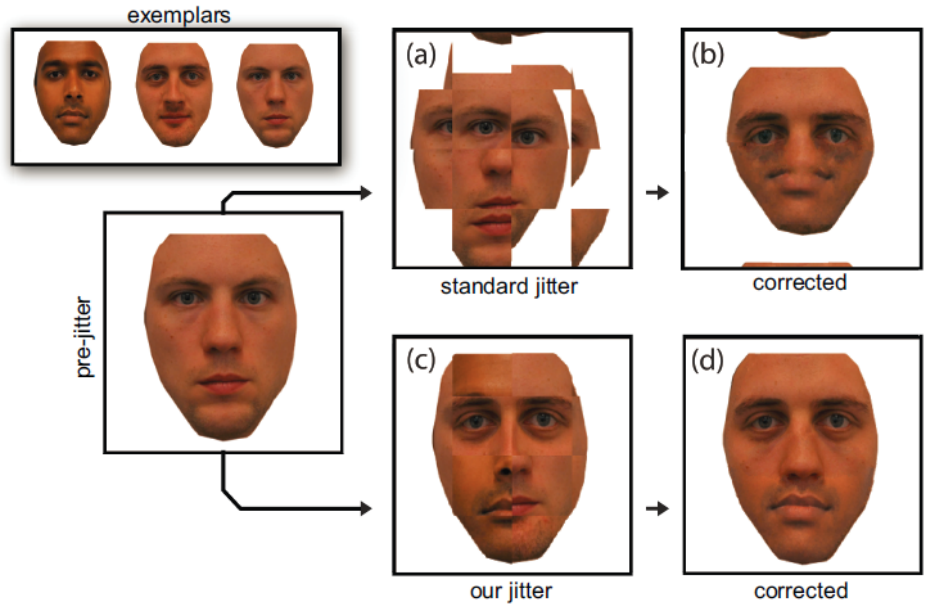


Figure 2.14: Structure preserving jitter of Risser *et al.* [Source: [4]].

(a)

(b)

Figure 2.15: An example of synthesizing the structured image hybrids from Risser *et al.* [Source: [4]].

to the globally-coherent input image (Tappen and Liu [175]). Hence, these methods do not explore the synthesis of highly-deformable or non-structured visual objects, as the alignment of such visual objects is difficult.

2.2.2 Appearance Image and Index Map

In a way, the problem of texture synthesis could be inverted. Assume that the target image was generated from a small appearance image and an index map that has the same size as the target image. So, the problem is to estimate the index map and the appearance image (palette) from the target image or a collection of images, such that the reconstruction is as accurate as possible. Jojic *et al.* proposed the Epitomes approach [176] and the Probabilistic Index Maps approach [177] to solve this problem. Later, an alternative approach called Jigsaws was proposed by Kannan *et al.* [178], with the principle illustrated in Figure 2.16. These approaches are probabilistic unsupervised methods of explaining away the target images with index maps and appearance images, with different constraints on the index maps and appearance images. This enables accurate estimation of the appearance and the position of image parts in the image. However, the spatial relation between parts in different images is not explicitly captured. Moreover, the parts in the appearance image may not match each other after running the algorithms on multiple images. The long run-time remains an issue for practical uses of the algorithm. The learned index maps and the appearance image successfully approximate the data, however, sampling this model for generating new instances of the visual object was not explored.

This kind of representation was used for a novel video editing tool [179]. Similarly, the image formation process was modelled as a set of transformations from the appearance image to the frames of the video. This representation was termed "unwrap mosaic". Hence, editing of the appearance image would propagate the edits to all of the frames of the video. In this work, there is no constraint on the compression of the appearance image; on the contrary, the appearance image is usually greater than the size of a video frame. Furthermore, the 2D transformations are smooth, except for the occlusions. This work relies on good tracking (optical flow) and cannot be readily applied to a set of images of different instances of some visual object.

2.2.3 Retrieval and Compositing

Numerous works proposed to expoloit very big image sets. A method that leverages Big Data was applied to the image inpainting problem [180], where

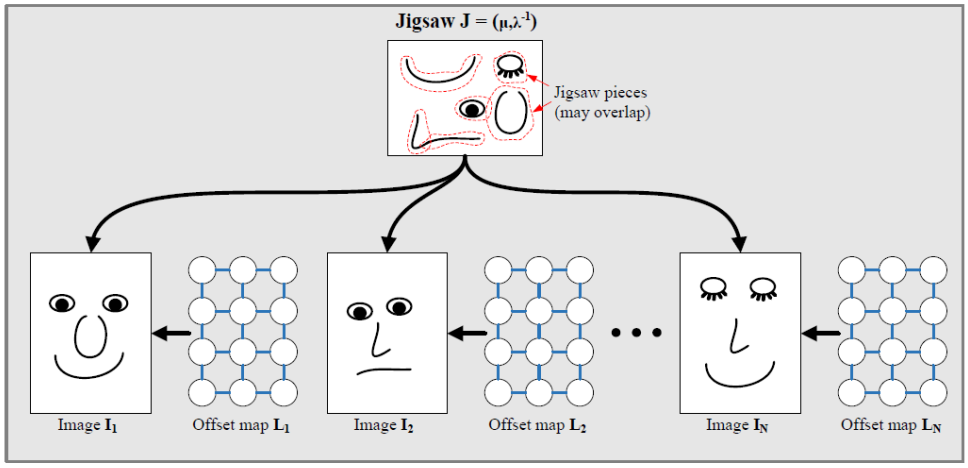


Figure 2.16: Illustration of the Jigsaws model of Kannan *et al*. Each image $\mathbf{I}_i$ is generated by copying over pixels from the Jigsaw image $\mathbf{J}$ indexed by the offset map $\mathbf{L}_i$. [Source: [178]].

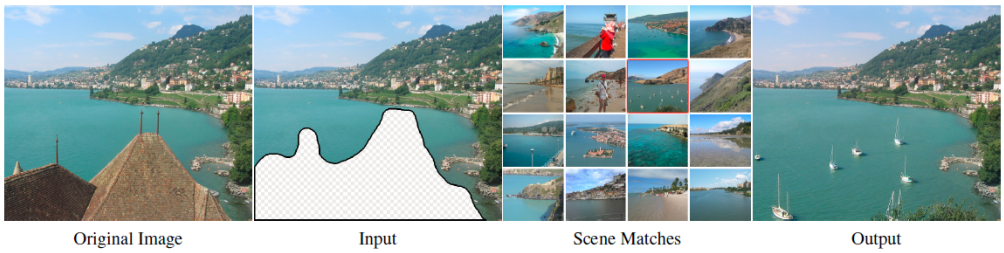


Figure 2.17: Example of scene completion of Hays and Efros. [Source: [180]].

the problem was reposed as identifying images that are likely to have similar pixel values in the regions that are missing in the input image. This approach exploits the vast amount of images in the internet. Since the number of images is in millions, a very basic method for matching images is used. Descriptors that incorporate gradient and color information are used to find nearest neighbors to the query image. The missing region is copied from the retrieved images and blended using Poisson blending. See Figure 2.17 for an example.

The main idea of using big numbers of images has also been exploited to enhance the look of the computer graphics generated scene [181], modify objects in an image [182], add objects into a scene from other images [183] or 3D models [184], create scene collages [185, 186], create photomontages from the sketches and some text keywords [187, 188, 189], or with the sketches only [190, 191, 192, 193], *etc*. Some of these methods use the search results from Google Images or equivalent image search engines and use the saliency-

based image cues for segmenting out the objects from the images [194]. Other methods use the labels of polygonally segmented objects provided by databases such as LabelMe [195]. Furthermore, it is possible to use 3D models to render new objects into a scene [184, 196]. One of the key components of such systems is the descriptor that transforms the user's input into the appropriate query. While the results look impressive, these methods work on the level of scenes and cannot synthesize novel examples of the visual objects.

An interesting application of a big database of images was used by Lee *et al*. [197] to provide a visual feedback to the user in an interactive fashion, so that the user can freehand sketch a drawing with a guidance from the system. The database was compiled by downloading images from Google Images related to different query words. It is assumed that among retrieved results on average there are a significant number of images that the user may want to draw, and the false results are rare. The images were converted to edge maps, each one subdivided into cells in a grid. Edge-based descriptor was used to match strokes drawn by the user to the cells in the database. The visual feedback, so-called "shadow", is an aligned average of images that have cells corresponding to the user's input. Although, the sketches drawn with the help of this system have more realistic proportions than unassisted drawings, the outputs are only sketches (grayscale edge images), not realistic color images.

Even when the number of images is not big, it is still possible to query the images for interactive systems that create photo-montages [198]. Another approach is to use existing image as a query. For example, Kemelmacher-Shlizerman *et al*. [199] use an image of a person A to query a collection of images for an image of a person B with a similar face pose and expression. This allows to control a face image of a person B using video input from a camera. In another work, Kemelmacher-Shlizerman *et al*. [200] create face animations that display images from a collection of images of some person by cross-blending between images. Dale *et al*. [201] proposed a system of face puppeteering, where a video sequences of a face talking is used as a target to a face model; the system computes face pose and expression descriptors for each frame; then queries source video of another person for frames that have corresponding face poses; and replaces face region in the original video with the face from the source video. Yang *et al*. [202] use collection of face images of the same person to correct a target image's expression. This is done by warping target image so that the expression matches source image from the collection.

2.3 Combining Parameteric and Non-parametric Methods

Finally, we will review works that combine the global and the local methods. These methods work in two steps. First, the global model is used to constrain the result to have global coherence. To eliminate artifacts inherent to the global model, such as absence of the high-frequency signal, the second stage searches for the patches that are close to the global model's output and are coherent with the neighboring patches. Then, these patches are blended to produce the final result.

Mohammed *et al.* [1] has combined the global and the local models in "Visio-lization" method for the face synthesis problem. The factor analysis was used to train a global parametric model for faces. Random samples from this subspace produce globally-coherent face images, but lack detail (see Figure 2.2). For the second stage, an approach similar to the image quilting was used to hallucinate details. The library of images was searched to find patches that matched low-resolution sample of the global model. Then, these patches were blended using Poisson image editing approach [203] to produce the final result. These steps can be seen in Figure 2.19. Results for both the profile and the frontal face images were shown, although the identity was not preserved. Mohammed *et al.* [1] demonstrate various applications of the combined model: editing faces, changing expressions, changing hairstyle, inpainting, *etc.*

It is important to note that the global model is necessary to produce results that have global coherence. Otherwise, the face may change gender, hair color, *etc.*, as can be seen in Figure 2.18.

Although, Mohammed *et al.* [1] demonstrate simultaneous synthesis of frontal and profile views of the face, ensuring the agreement of details between different views is not explored.

Similar approach was proposed for the face hallucination problem by Liu *et al.* [204, 205]. Since the problem is already constrained by the input low-resolution image, the global model is used to generate upsampled version of the input. For global model PCA jointly models the low-resolution and the high-resolution images. In order to reduce artifacts, this global model preserves a lot of dimensions. To complement for the high-frequency signal, a Markov Random Field is used to find optimal patches that minimize the distance between low-frequency signal of the patches and the global model's output. The model is illustrated in Figure 2.20.

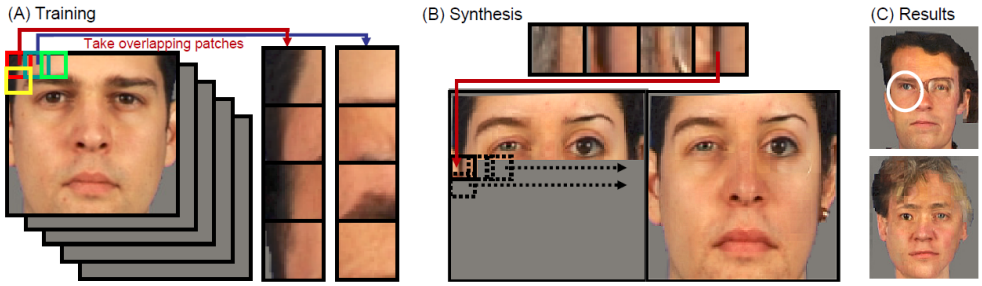


Figure 2.18: Generating faces with the local coherence and coordinate constraints only. The algorithm blends patches from images of different individuals which produces results without global coherence. [Source: [1]].

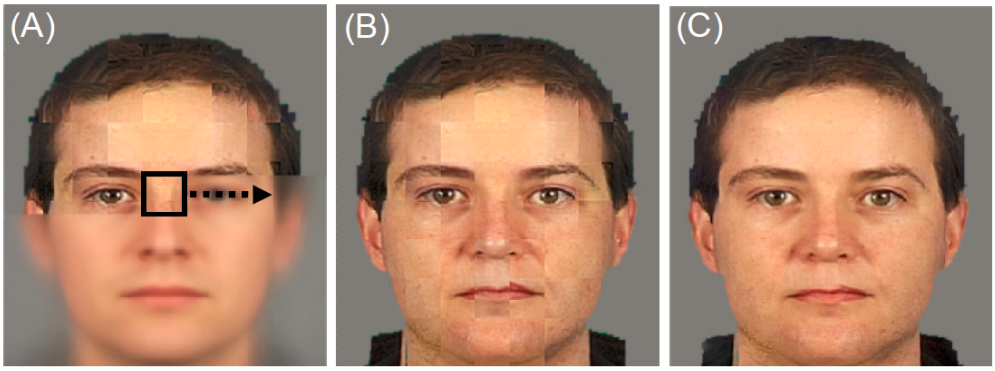


Figure 2.19: Adding the high-frequency signal to the output of the global parametric model. [Source: [1]].

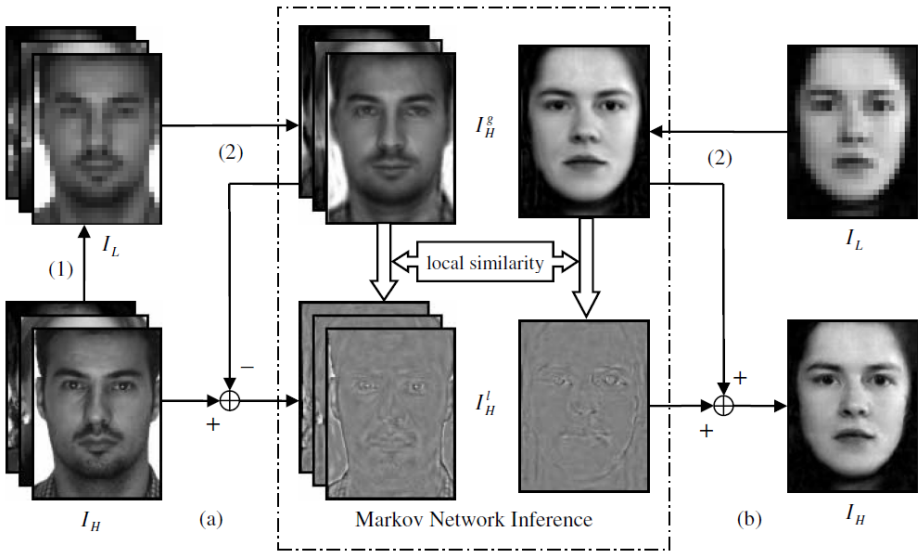


Figure 2.20: Face hallucination framework by Liu *et al.* [Source: [204]].

Overall, Liu *et al.* [204, 205]'s method has similarity to "Visio-lization" [1], but solves the hallucination problem, rather than synthesis problem. Consequently, sampling of [204, 205] does not generate good face images as compared to "Visio-lization" in [1].

Mortazavian *et al.* [206] also combine global and local models to solve face hallucination problem. They use Morphable Model [49], a 3D global model, to allow arbitrary rotations of the input image. However, their local model [207] still operates in 2D space of the unwarped texture map. Their results are comparable with Liu *et al.* [204, 205], but allow arbitrary views as input.

Recently, Dessein *et al.* [208] proposed to combine global and local models that both operate in 3D space, rather than in 2D space to solve the problem of super-resolution or detail hallucination of faces. So, Dessein *et al.* use Morphable Model [49] as a global model to align the input image with the library of images. This allows to model faces under arbitrary rotations and views. Furthermore, the method uses texture patches on a 3D mesh, rather than 2D image, which allows to use multiple images under different views as input to the method. Moreover, the resultant 3D model can be rendered in any pose or view. The generated results in the problem of super-resolution and texture completion are impressive, especially for low-resolution images of 12 pixels wide (*i.e.* downsampling factors of 16). However, Dessein *et al.* use the global model to align the input image and the library of images, so that the result is conditioned on observed data, rather than sampling of the global model to synthesize novel faces.

2.4 Summary of Related Work

We have shown relevant works in the area of synthesizing novel instances of visual data. The related works were presented in three sections: parametric models, non-parametric models and combined methods.

For non-parametric models, a wide range of methods was reviewed. Among them, Image Hybrids formulate interesting approach. Since the number of input exemplars is very small, the risk of "drifting away" (see Figure 2.18 (B) and (C)) from plausible output is reduced, as long as the exemplars are close semantically. In Chapter 3 we propose to specify exemplars of articulate objects through interactive sketching and extend Image Hybrids synthesis approach to highly-deformable visual objects through semantic part labelings. Furthermore, the user interactions in the synthesis process are also consider in the design of the system.

The parametric approaches were broadly visited, with the analysis of the advantages and the drawbacks. As previously mentioned, there is no global parametric model that can reliably model structural changes, occlusions, severe pose changes and articulations. In Chapter 4, we show that it is possible to formulate more powerful global model that can cope with structure inconsistencies by conditioning the pixel intensities on local structural contexts.

The combined methods cannot be readily applied to synthesize images from datasets of highly-deformable and non-structured visual objects. Firstly, the parametric models cannot model visual objects with significant deformations, occlusions, croppings and varying structure as we discuss this issue in Chapter 4. Secondly, lack of alignment of images implies that the search for a matching patch cannot be done over a region in each of the images, but rather the matching patch search has to consider all patches from all images of the dataset. This is a challenging problem, which is investigated in Chapter 5.

Chapter 3

Interactive Sketch-Driven Image Synthesis

"All you need to paint is a few tools, a little instruction, and a vision in your mind."

—Bob Ross

In this chapter we present an interactive method for composing realistic images of an object under arbitrary pose and appearance specified by sketching. Our method draws inspiration from a traditional illustration workflow: The user first sketches rough "masses" of the object, as ellipses, to define an initial abstract pose that can then be refined with more detailed contours as desired. The method is made robust to partial or inaccurate sketches using a reduced-dimensionality model of pose space learnt from a labelled collection of photos. Throughout the composition process, interactive visual feedback is provided to guide the user. Finally, the user's partial or complete sketch, complemented with appearance requirements, is used to constrain the automatic synthesis of a novel, high-quality, realistic image. We evaluate our method with two user studies.

3.1 Introduction

Suppose you would like to create a realistic image of an animal – a horse, for example. You can imagine the horse's pose and appearance in your mind's eye, but how can you translate that vision to an image? Painting realistic images with correct proportion, pose and color requires training and talent that most of us lack. Thanks to Google and other search providers, it has become easier for novices to search the internet for images of a given object category using keyword search. But finding examples in a particular pose or relationship

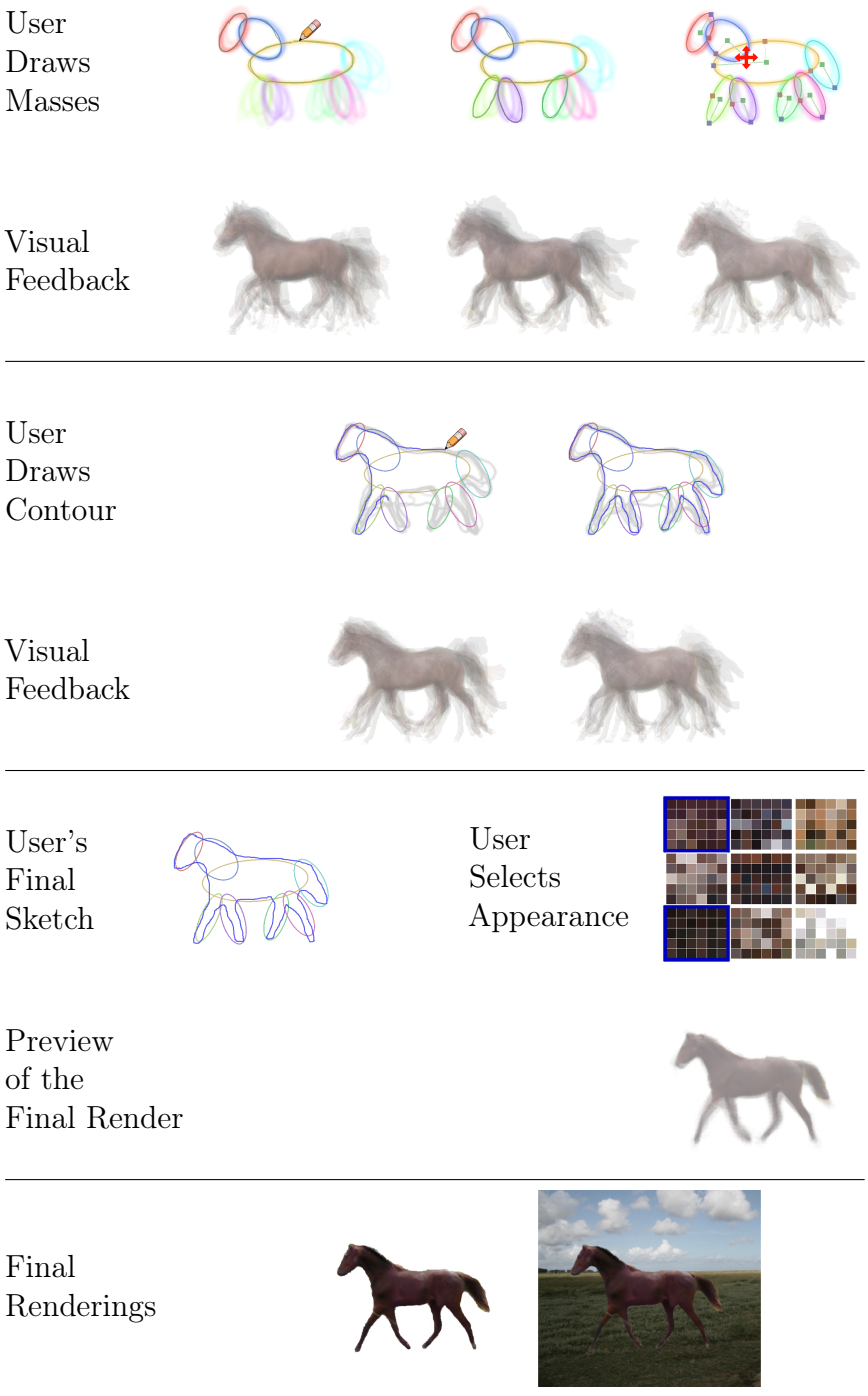


Figure 3.1: Our interactive method guides a user to specify pose and appearance using sketching in order to synthesize novel images from a labeled collection of training images. The user first sketches elliptical "masses" (top), then contours (middle), mimicking a traditional sketching workflow. Once the pose is specified, the artist can constrain the appearance and render a novel image (bottom). In each section the top row are user sketch input and feedback guidelines; and the bottom row are rendered previews.

to the camera requires searching through pages and pages of search results. Furthermore, it is possible that no single image matches what you originally imagined: perhaps, one image almost has the correct pose, another has details you like, and a third has the color you had in mind. Combining and modifying these images to produce the one matching your goal is very involved, even with state-of-the-art image editing software.

In this chapter we present an end-to-end method that enables users to interactively sketch and synthesize novel images, given a database of labelled images. When designing a method for controlling image synthesis, a critical challenge is defining user interactions that are meaningful to a human, but also express appropriate constraints on the synthesis. To address this challenge, we draw inspiration from the strategies employed by figure drawing artists: it is common for such artists to begin by sketching the rough body proportions using overlapping ellipses or other simple abstract shapes, often called *masses*. Once the masses are sketched as desired, the artist can go on to add contours, shading and finer details of the figure. While this serves as a useful starting point, it is not sufficient to serve non-expert artists: We cannot expect a casual user to produce a sketch that resembles a realistic image as input to a synthesis engine without assistance. Therefore, the method must guide the user in an exploration of the synthesis space rather than simply optimizing a set of pre-defined constraints.

More concretely, we propose that a sketch-driven image synthesis method must meet four intertwined design principles: First, it must be *responsive*, providing results rapidly enough for a user to iteratively refine her mental concept. Second, it must be *exploratory*, guiding the user through the space of possible syntheses with meaningful feedback. Third, it must be *robust* to missing data, providing meaningful feedback in the face of incompletely specified constraints. And fourth, the interactions should be as *fluid* as possible – preferring sketch gestures over menu or label selections.

These principles have driven all of the design choices for our method: In the initial "mass" phase we support freehand sketching of new masses. Our method supports traditional "overdraw" to adjust these masses after their initial placement, but we also provide direct manipulation of mass ellipses. A preview is shown at all phases, which increases in specificity and concreteness as the constraints are refined. The preview is updated after each sketch gesture, rather than waiting until the sketch is complete. In early stages this takes the form of blended images from the database, giving a visual representation of the local

space that meets the constraints specified thus far.

We also provide feedback in the form of shadowed lines underneath the user's sketch. These shadows suggest, first, suitable ellipse locations and, second, valid object contours. The recommendations are derived from a probabilistic model of the labeled body-parts in the image database and help an untrained user to draw masses and contours in appropriate layouts for each object. We adapt existing techniques to synthesize a final image that is consistent with the user's sketch.

The masses in our method are used for two purposes: they serve as a proxy for specifying pose and they are used to guide the final synthesis stage.

Artists use a variety of geometric primitives as masses: The human pelvis is often represented with a cuboid, and arms with cylinders. Human faces can be represented using ellipsoids sectioned at various ratios along their axes (similarly to reference lines in [209]). Although our method could be augmented to support numerous primitives, we focused our initial efforts on a single one – the ellipsoid. Even though it is not commonly used in sketching humans, this primitive is flexible enough to sketch a variety of walking and flying animals, and thus our datasets span such animals. Incorporation of other common parametric primitives such as rectangles, conics or generalized cylinders [210] is left as future work.

To our knowledge, ours is the first image synthesis method that interleaves sketched constraints and preview synthesis. We argue that this interaction modality facilitates joint human-computer exploration of a space of synthesized images, and is thus the foremost contribution of our work.

In addition, we apply machine learning techniques to learn a low-dimensional manifold from the data that models the joint configuration of masses and the contour shape of objects, a highly complex relationship that could not be specified using a heuristic approach. We bring together appropriate representations for the masses (Stokes parameters [211]) and the contours (elliptical Fourier coefficients [212]) to ensure that the manifold interpolations are plausible even when using relatively few input samples, unlike ShadowDraw [197] which requires a dense sampling of each sketch configuration space. For the final synthesis stage, we adapt the Image Melding algorithm [159] by using additional guiding layers corresponding to each distinct body part and the silhouette.

3.2 Related Work

Some existing systems, such as Sketch2Photo [188] and Johnson *et al.* [187], have merged retrieval and synthesis into unified systems. Both allow a user to sketch a query combining text, images, and/or outlines; retrieve matches from the internet or a local database; and *compose* a new image using the retrieved elements. Goldberg *et al.* [182] use the Sketch2Photo framework to allow object-level manipulation in images using online images queried by the user's keywords and segmentations. They propose novel deformation and alignment techniques to achieve high-quality results. However, neither method is targeted for interactive use: Johnson *et al.* report 15 to 45 seconds per composition, Tao *et al.* report 15 minutes for each object retrieval and another 5 minutes for composition; and Goldberg *et al.* report 10 minites for object retrieval. We posit that a more interactive system is critically important for creative processes, allowing users to quickly explore the space of possible outcomes.

Photo clip-art [183] and Photosketcher [190] are two other systems that are designed for interactivity. Photosketcher, in particular, uses sketches as input. Similarly to Photosketcher, Sketch2Scene [184] uses user sketches to retrieve and place 3D models to assist the user in the 3D scene modeling from a dataset of 3D models. However, like the other methods, these systems *compose* the final output using only one image or 3D model per object rather than synthesizing new objects using combinations of retrieved images. Furthermore, none of these previous approaches provide a mechanism for detailed pose specification.

Recently, the PoseShop [189] system proposed using online image search to construct a segmented human image database with pose and semantic action descriptions. PoseShop queries the database with a sketch or skeleton, allowing a composition of personalized images and multi-frame comic strips by swapping the head and clothes to the user's specifications. This system does compose novel images, but does not provide interpolation between poses and uses only one database image for each output.

Other recent works blend elements from a collection of related images. For example, Mohammed *et al.* [1] learn a global parameterized model of frontal face images, and constrain a patch-based texture synthesis algorithm using a sample from this model. Risser *et al.* [4] demonstrated a hierarchical pixel-based texture synthesis algorithm that generates novel image hybrids by jittering exemplar coordinates instead of spatial coordinates in order to preserve structures. However, neither of these methods supports pose variation

or provides direct user control of the synthesized result. The PatchNet method combines multiple input images for each output, taking into consideration contextual relations between parts [161]. However, it was only demonstrated for scene composition rather than object posing, as it does not incorporate an explicit model of pose. Recently, Darabi *et al.* [159] demonstrated a patch-based method called "image melding" to smoothly interpolate textures and colors across images. Our synthesis engine builds on the image melding framework with additional control channels, as in image analogies [171].

In the domain of drawing assistance, Lee *et al.* recently proposed a system that allows freeform drawing of objects [197]. As the user adds strokes, the system interactively provides shadow-like hints of where the next stroke should be. The system does not have any prior knowledge about the object that the user is drawing, so the shadows are constructed by blending relevant edge-maps from a large database queried using local edge patch descriptors. Sketch-Sketch Revolution [213] allows novice users to learn to replicate strokes of an expert artist by following guidance and feedback of the system in a step-by-step tutorial, which was previously created by an expert artist. The iCanDraw? [209] system assists users by generating *corrective* feedback extracted from a reference image. Similarly, The Drawing Assistant [214] automatically extracts *"block in"* visual guides from a single reference image and provides corrective feedback to the user. Sketches produced with the help of these systems have more realistic proportions than unassisted drawings, but the outputs are only contour sketches, not realistic images. Inspired by these systems, we strive to utilize sketch inputs to produce plausible realistic image outputs, restricting our attention to the case in which the class of object is known in advance.

3.3 Sketch Interaction

One traditional sketching method is to first draw the outline of the subject using primitive shapes likes ellipses, circles, squares, *etc.* [215, 216, 217, 218]. At this stage only the gross relationships of body parts are specified, and the simplicity of the shapes makes it possible to adjust and iterate quickly. It is important that these shapes are very basic, allowing the artist to define the correct proportions between parts of the object without focusing attention on small scale details. Once the rough outline of the object is set, the artist can add finer details. Beyond this point, the masses act as an anchor for the drawing so that the artist can focus on local details without breaking proportion or symmetry. An example of this approach is given in Figure 3.2, showing an

artist using masses to sketch a horse and a pigeon. Our goal is to mimic this approach by providing visual feedback that can guide the user in adjusting masses and defining strokes.

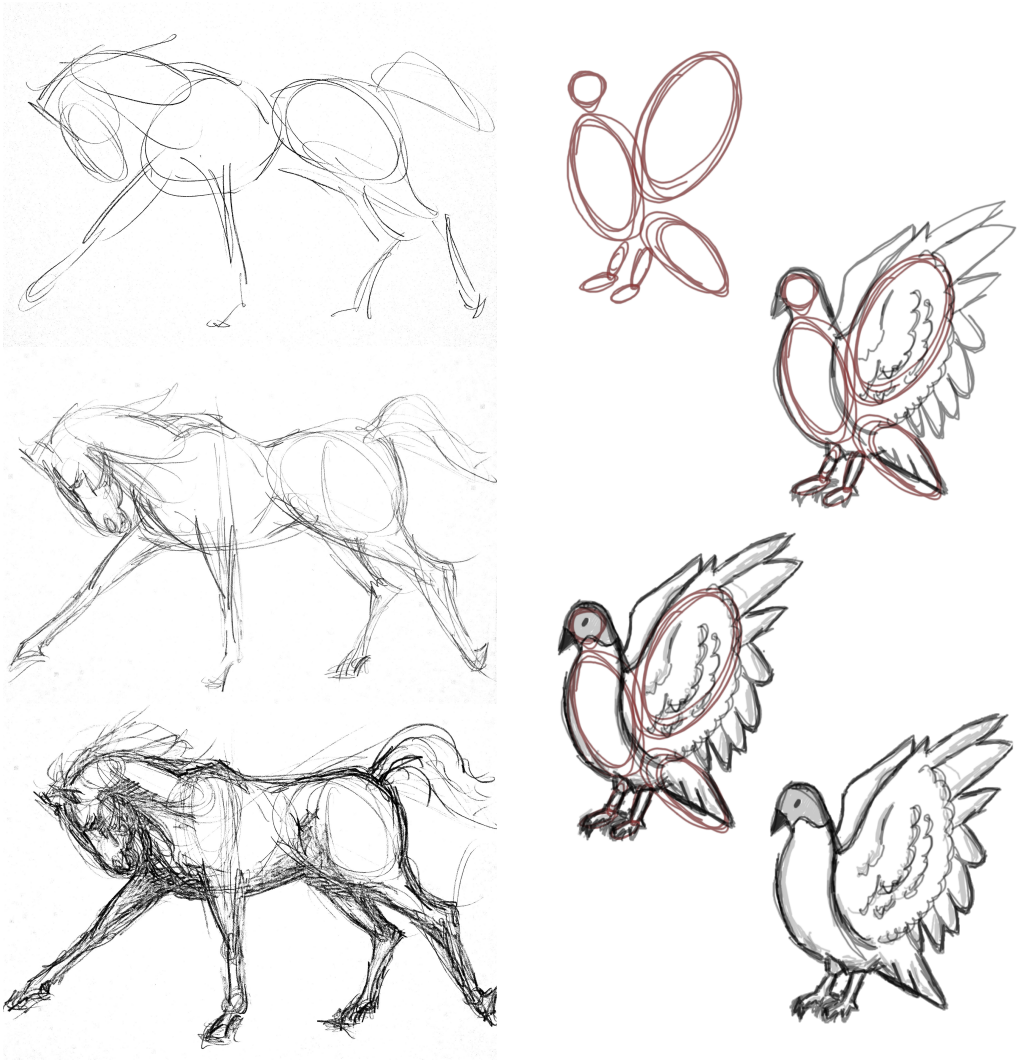


Figure 3.2: Sketching a pigeon and a horse by hand using masses. The artist starts by drawing in the masses for the body parts in the correct proportions before continuing to fill in the contours and then any final details, such as shading.

We argue that the use of elliptical masses is a more effective tool for specifying gross shape than either contours or "skeletons." Contours contain both small and large scale detail, and it is typically difficult for a novice user to focus on both scales at the same time as they trace an outline. Skeletons or "bones" are an appealing alternative due to their ubiquitous use in 3D computer graphics. However, whereas a line can express only (2D) length and angle of a body part, an ellipse can also express its apparent thickness.

Perhaps more importantly, novice users are not necessarily good at evaluating the proper location of "bones" within a figure. For example, in Figure 3.3, where would you draw a straight line specifying the location of the neck (cervical vertebrae)? Most people without veterinary training will be surprised to see that when a horse's head is raised, the neck vertebrae of a horse are closer to the front of the neck at its base, but closer to the back of the neck at its apex. In contrast, elliptical masses require no knowledge of internal anatomy, and they form a visual guide for the subsequent stage of contour sketching, since the contours often follow close to the mass edges.

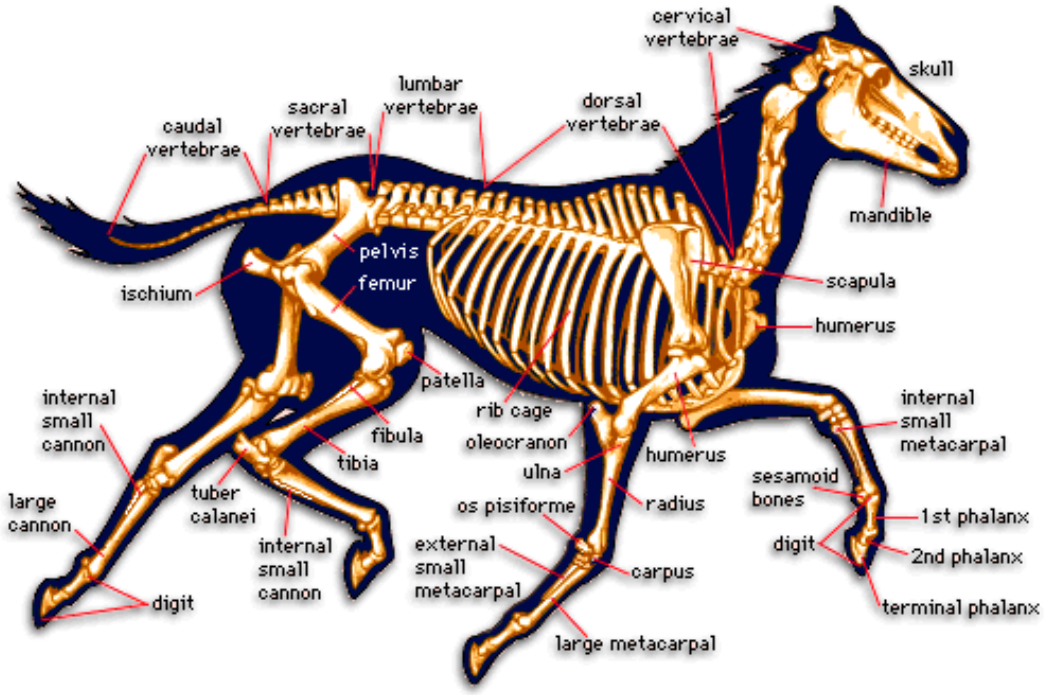


Figure 3.3: A cross-section of a horse showing that the location of bones is unintuitive: Many people are surprised that the neck vertebrae are closer to the front of the neck when a horse's head is raised. (By courtesy of Encyclopaedia Britannica, Inc., copyright 1998; used with permission.)

3.3.1 User Interaction

Our system's window shows two panels at all times: On the left, the *sketch* panel shows the user's sketched strokes, as well as semi-transparent guidelines suggesting possible stroke locations. On the right, the *preview* panel visualizes the space of possible output images, given the sketch progress thus far. Since the outcomes are increasingly constrained throughout the sketching process, the contents of both panes are constructed differently at various stages. The

entire workflow is given in detail below, with descriptions of the panel contents seen at each stage.

Initial State The user is first shown an abstract overview of the range of pose and appearance of the target object. The sketch panel shows guidelines for the elliptical masses, indicating this is the first step in the sketching process. Since the range of masses can overlap a lot, the guidelines for each part are shown using a different color to improve visibility and comprehension. They are slightly blurry, to emphasize that they are only loose constraints: The user can sketch anywhere, but results are best when the sketches are close to the range of real object variation. The preview panel is empty, since the output is totally unconstrained at this stage.

Drawing Ellipses The user paints strokes in the sketch panel (in grey). After each stroke, the sketch panel shows ellipses (with the color of the estimated part) fitted to the strokes. The guidelines are updated to show plausible mass configurations similar to the user's sketched masses. The preview panel shows corresponding nearest-neighbor images, blended together using simple averaging, giving a ghosted view of possible outcomes (we call this "Fast NN Preview").

Mass Adjustment The user can adjust existing masses in two ways: Those familiar with traditional sketching may prefer "oversketching," simply drawing over the previous strokes to replace them. However, novice artists may prefer to adjust the mass ellipses using an object-oriented approach. For these users we provide an adjustment mode in which the ellipses can be directly manipulated: Colored handles appear on the major and minor axes of each ellipse in the sketch view. Dragging the center handle translates the corresponding ellipse, and dragging the axis handles rotates and/or scales it about the center. The preview panel is constructed in the same way as the previous paragraph.

Drawing Contours In this mode, the sketch panel shows faded contours of real images similar to the user's sketch, much like ShadowDraw [197], but interpolated using our manifold model.

Editing Appearance The previews described in the previous paragraph blend together multiple input exemplars, and can thus obscure specific appearance details such as color and lighting. To address this, we also provide an appearance selection mode in which the preview window switches to a grid of color palettes computed from the database exemplars that are near-matches to the current sketch. The user may click on one of these palettes to constrain the output appearance, and then return to sketching. The preview panel now shows

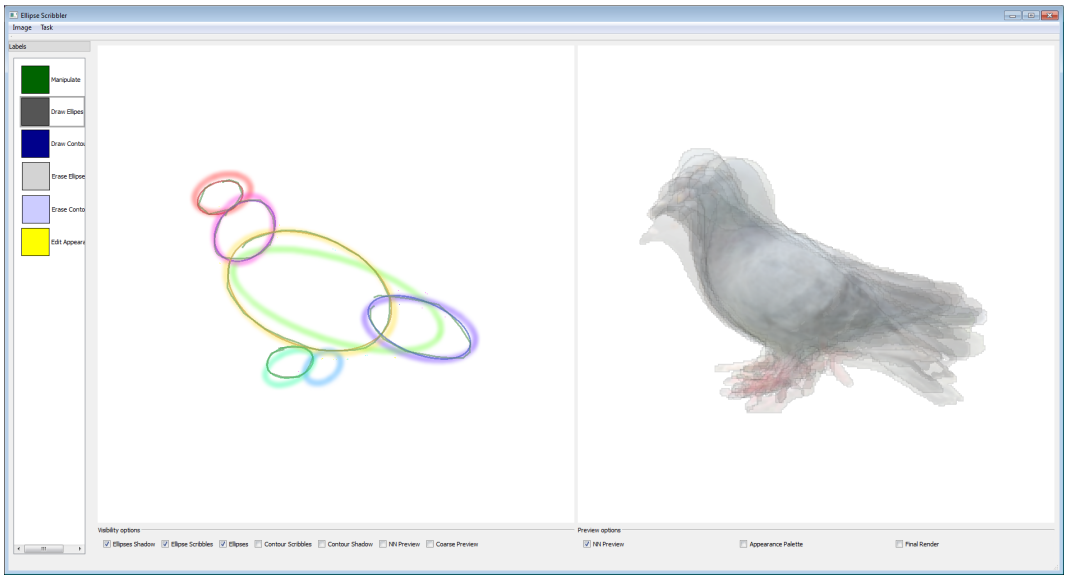


Figure 3.4: Our system's interface. The left panel is the user's drawing canvas, where the shadow feedback is shown. The right panel shows the fast nearest-neighbor preview.

a fast low-fidelity synthesis result aligning the images to match the sketched contour.

Final Synthesis When the artist is satisfied with the constraints and preview render, she can request a final rendering, which may take 3-4 minutes depending on the resolution of the images in the dataset. We use the user specified contour and the ellipses to guide the synthesis process, as this additional information helps the system deform the images of the dataset before blending them together.

Although we present the steps above in their logical sequence, the system does not require a strict linear progression through these stages: The modes can be revisited in any desired order. The artist may choose to constrain color before pose, or return to mass adjustment after drawing part or all of the object contour. Furthermore, the user can select appropriate visual feedback for each of the interactions.

Reflecting back on the four design principles proposed in section 3.1: Our system is *responsive*, as both panels are updated after every user stroke. It is *exploratory*, because it attempts to illustrate at each stage the span of plausible outcomes given the current sketch. It is *robust*, by virtue of providing this feedback after a single user stroke, or hundreds. And it is *fluid*, utilizing hand-drawn strokes wherever possible; discrete menu or tool selections are required at only a few moments in a typical interaction. Figure 3.1 illustrates

some stages from our workflow, and Figure 3.4 shows our system GUI running on the pigeon dataset. Please see Appendix B for screenshots of our system GUI with different visual feedbacks in Figures B.1 to B.7.

3.4 Implementation

In this section we describe details of the technical approach taken to provide the interactive workflow of the previous section. The most significant challenge is to provide a joint model of object pose, contour, and appearance space that a user can explore continuously and freely, within the constraints of plausible image synthesis. In addition, the system design is made more challenging by the requirement that everything must run at interactive rates, with the exception of the final image synthesis.

We propose an image melding [159] approach to synthesize novel images of a particular object: Our method combines multiple images of similar objects under similar poses to produce a final image in a specific pose. To allow for a wide range of poses and appearances of a particular object, we require a database of images containing differing poses and appearances. The prevalence of online image search and image libraries renders such an image database straightforward to obtain.

Once we have a collection of images to use for synthesis, we must consider how to address the fundamental requirements of our interactive workflow:

1. How do we identify reasonable configurations of masses to guide the user when specifying pose?

2. How do we identify feasible object contours given the pose of the object that will allow for accurate synthesis?

3. How can we inform the user of the possible variation in appearances of the object?

4. How do we select appropriate images to use to synthesize this specific pose and appearance?

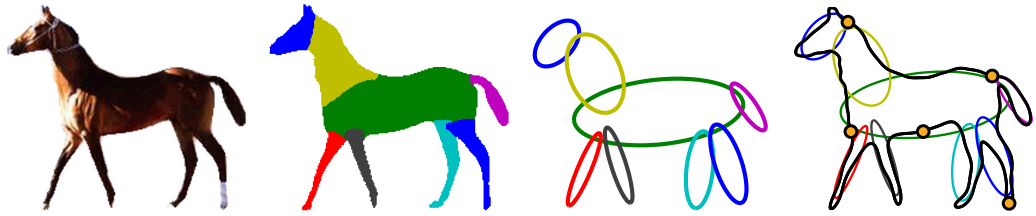
We adopt a single methodology to deal with all of these questions; we make use of machine learning approaches to model the joint relationships of mass pose, object contour, and the training images (which encode object appearance) in a unified probabilistic framework. More specifically, we optimize, within the high-dimensional joint space of mass poses and contours, a low-dimensional manifold that contains all of the database images.

Using appropriate parameterizations, we can move continuously within the manifold, and smoothly interpolate the masses and contours between training images in order to generate valid novel poses. A location on the manifold identifies a specific pose, and the nearest neighboring database images in the manifold are good candidates for image synthesis. In addition to answering the above questions, the probabilistic nature of the model also allows us to handle image synthesis in the presence of incomplete information, *i.e.* missing masses or contours, affording our method a degree of robustness.

In order to train the model, we have to provide labeled data to associate the poses of each mass and the contours with each image in our database. This consists of segmenting each image into a set of body parts. From this segmentation we may fit a set of ellipses (as the object masses) and find the contour of the object.

Given this framework for feedback and synthesis, we must also consider the user input. User scribbles are interpreted as editable masses, allocated to specific body-parts, and the contour and appearance constraints must be specified. We now visit each specific component in further detail.

3.4.1 Training Data



(a) (b) (c) (d)

Figure 3.5: Example of training image segmentation. (a) The input image is segmented from the background then (b) split into its constituent parts to allow (c) ellipse fitting to represent the masses. (d) The contour of the complete silhouette and the alignment key points are then automatically extracted.

We require a collection of images (dataset), containing a single class of objects, where the object of interest is segmented from the background. For each object, we define a model describing how the object is divided into masses. For example, a horse can consist of head, neck, torso, tail, left and right forelegs and hind legs.

So, we have compiled four datasets:

- We have purchased a photomontage of cats captured on a white background by professional photographer. The photomontage has 392 images

of Cats. The license does not allow sharing of the photomontage, so we can only publicly show derived results. The images were segmented from the white background and manually labeled. The labels are: head, torso, tail and 4 legs.

- We downloaded 281 images of Elephants from internet, manually segmented them, and labeled with following labels: head, torso, 4 legs, trunk.

- We downloaded 270 images of Pigeons from internet, manually segmented them, and labeled with following labels: head, neck, torso, wing, 2 legs, tail.

- We have labeled 328 images of Weizmann horse database [219, 220, 221] that already has segmentation masks. The labels are: head, neck, torso, 4 legs, tail. We removed duplicates and images with severe visual artifacts to a reduced set of 295 images.

All images were flipped, so that the animal is looking in the front to left direction. Examples are shown in the Appendix A. We labeled each image by assigning each pixel to the relevant part of the object model. We then extract masses by fitting ellipses to the boundaries of each of the labeled parts. We chose ellipses to represent masses because they are popular among artists [218, 215], it is possible to fit them to curves efficiently [222], and their shape is general enough to approximate many body parts. We note that any other shape with low-dimensional parameterization may be readily substituted in our method.

3.4.2 Joint Manifold

This subsection on the Joint Manifold was implemented by Neill Campbell. It is mentioned here as it is a critical part of the method.

To produce good synthesis results, we must ensure that the training images and the user-specified ellipses and contours are compatible. For example, the head of a horse cannot be placed on the far end of its tail. To achieve this, we provide feedback to users during sketching. This requires a statistical model of the joint space of ellipses and contours covered by our training data. Such a model can estimate the likelihood of a particular arrangement of masses and contours; this is then used as a measure of how readily such a configuration may be reproduced from the training data.

We require that the model be sufficiently powerful to represent the complex interactions between the ellipses and the contour, a multi-modal distribution, and also allow for fast inference queries to be performed at interactive rates. We achieve both of these goals by representing the contours with elliptical Fourier coefficients [212] and modeling the joint manifold of the ellipses and contours using a Gaussian Process Latent Variable Model (GP-LVM) [36]. We now discuss each of these components in further detail.

Representation In order to interpolate smoothly between contours in different training images, our method needs a continuous representation of the contour. However, the silhouette contours obtained from the segmented training images are not registered to one another with a dense correspondence. In any case, silhouettes from different viewpoints cannot be placed in a meaningful correspondence; for example, the front legs of a horse may appear separately or on top of one another. Inspired by the work of Prisacariu and Reid [223], we represent closed contours using elliptical Fourier coefficients [212], which can smoothly interpolate between the silhouettes of objects such as people, cars, and animals. Whereas Prisacariu and Reid used these silhouettes as a shape prior for segmentation and tracking, we will use it as a shape prior for image synthesis.

Fourier contour representations must be phase-aligned (*i.e.* a common starting point and parameterization) to achieve good interpolation [223]. We achieved high-quality interpolation by aligning a series of key points distributed over the length of the contour. This corresponds to resampling the contour such that there is a fixed number of samples between each key point. We compute a sparse set of key points on the contour using the labeled parts in an automatic fashion; for example, a point that lies on the leg and is farthest from the torso, and a point on the torso that is closest to the tail. These key points were chosen to be empirically consistent between different poses and mass configurations. For the horse dataset, we use five key points as shown in Figure 3.5(d).

The general parametric form of an ellipse is expressed by 5 values $\Big[x_c, y_c, a, b, \phi\Big]$: the x-axis and y-axis coordinates of the center of the ellipse, the length of the major and minor axii and the angle between the x-axis and the major axis, respectively. Here, ϕ is in the range $[0, 2\pi]$. However, we require a smooth representation for the set of ellipses for each training image. We

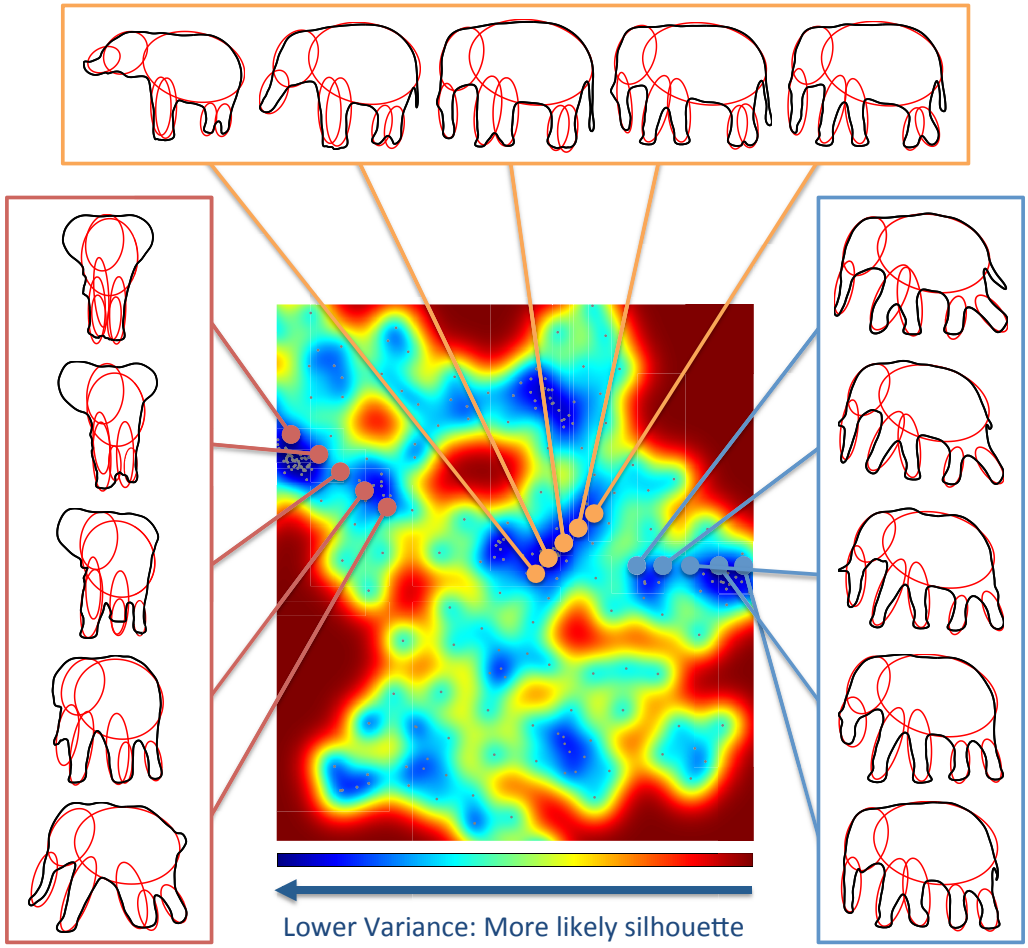


Figure 3.6: A 2D joint manifold of ellipses and contours learnt for the elephant dataset. Each point represents a configuration in pose space, and the color indicates the variance of the embedding in the latent space. Regions with a low variance are higher probability in the pose space. The location of the original training images are shown as grey dots.

achieve this using Stokes parameters [211] that are defined as

$$\left[x_c, y_c, a^2 + b^2, (a^2 - b^2)\texttt{cos}(2\phi), (a^2 - b^2)\texttt{sin}(2\phi)\right] \ . \tag{3.1}$$

Manifold Given the continuous representation, it is possible to interpolate between similar training images to generate new pose configurations of ellipses and contours. However, we cannot, in general, interpolate linearly in the space of Stokes and Fourier coefficients. Instead, we find a low-dimensional manifold in the joint space of ellipses and contours that contains the training image configurations, using a Gaussian process latent variable model [36]. We use

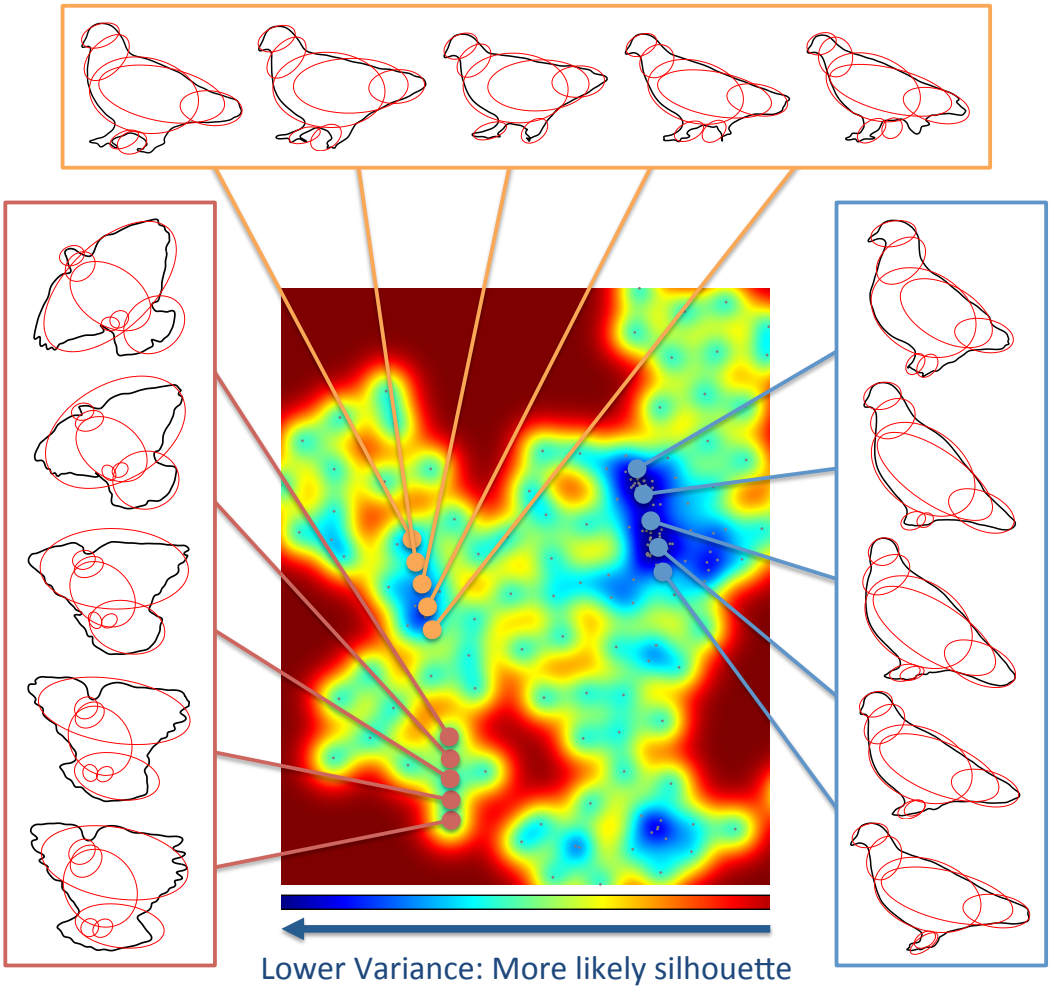


Figure 3.7: A 2D joint manifold of ellipses and contours learnt for the pigeon dataset. Each point represents a configuration in pose space, and the color indicates the variance of the embedding in the latent space. Regions with a low variance are higher probability in the pose space. The location of the original training images are shown as grey dots.

radial basis function to compute distances between data points in the latent space. In cases where body parts are occluded we will be missing some ellipses; hence, we employ the method of Navaratnam *et al.* [224] to train a GP-LVM joint manifold model with missing data.

The probabilistic nature of the GP-LVM model allows us to interpret the variance of the embedding in the low-dimensional latent space as the likelihood of a pose given the training data. Figures 3.6 and 3.7 show an example of the manifold learnt for a set of images of elephants and pigeons, respectively. The coloring of the manifold shows the variance of the ellipses and contours that would be estimated at that point. Hence, areas with a low variance (shown

as blue in Figures 3.6 and 3.7) are more likely to produce good results under image melding from the local neighboring training images (shown as grey dots on the manifold). The best dimensionality of the manifold depends on the training data. We experimented with different dimensionalities of the manifold and found that 2 and 3 dimensional manifolds generate best interpolations for our datasets, for simplicity we used 2D manifolds.

3.4.2.1 Ellipse and Silhouette Queries

We now provide details of how to provide the shadowed ellipses and contours used during the interactive sketching process. We first consider how to identify plausible locations for the remaining ellipses, given that the user has already drawn one or more of the masses. We then consider how to identify reasonable object contours given a set of masses and partially sketched contour fragments. Whilst both these approaches are used to provide visual guidance to the user sketching process, they can also be used to fill in incomplete data for the final synthesis (*e.g.* if the user has missed some of the masses or drawn an incomplete contour).

Ellipse Manifold Queries We define a cost function based on the difference between the query ellipses – which may be partially specified – and a sampled set of ellipses from the latent space; we use the L-1 norm in the Stokes parameter space. Since we cannot evaluate analytic gradients, we must perform this optimization using point estimates from the latent space (in a similar fashion to Navaratnam *et al.* [224]). The joint cost is multi-modal, so we start from a set of initialization points that span the latent space. This optimization can be performed rapidly, since the dimensionality of the latent space is so low and the ellipse cost function is very cheap to compute. We run the query, starting with around 60 initialization points (found by clustering the latent space points corresponding to the training images using k-means), in less than half a second with a two-dimensional latent space. Hence, the response to each query involves local optimizations with multiple starting points.

Figure 3.8 shows an example set of query ellipses and the six most probable modes, both as points on the manifold and in the form of the ellipses and contours. We also demonstrate that the training images with locations in the manifold closest to the mode have similar ellipse and contour layouts and are thus suitable to be used as source images in the synthesis stage.

In addition to the cost of each mode, which encodes how well the ellipses of the mode match the query, the manifold also returns a variance. This

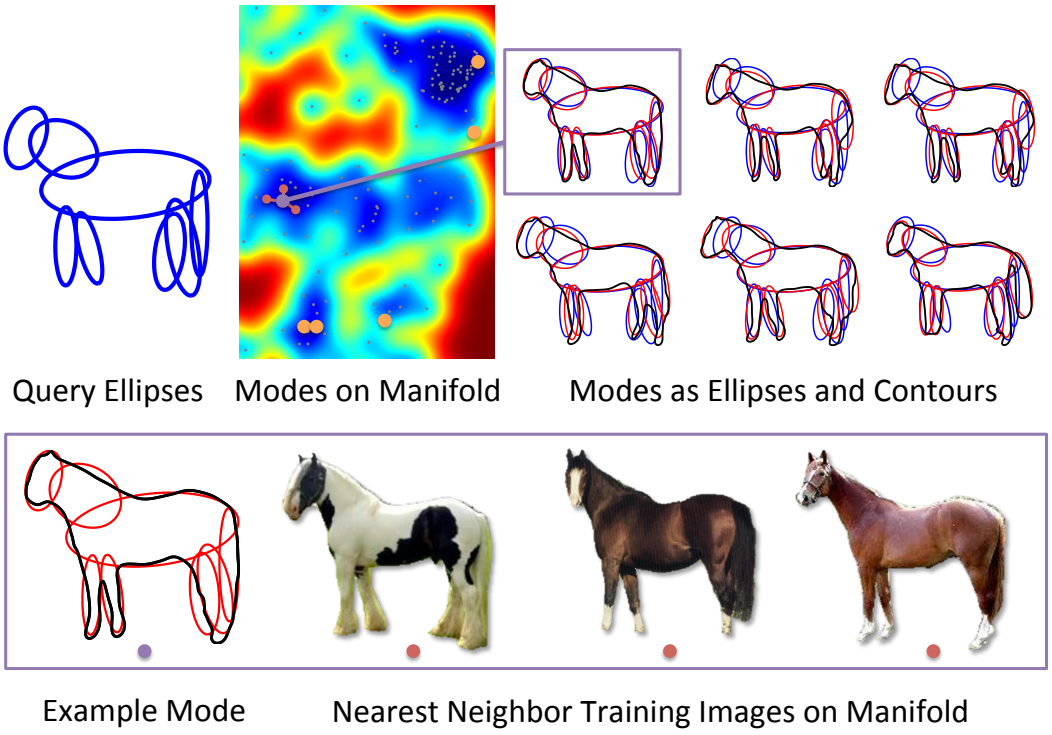


Figure 3.8: Upper left: a set of user-specified ellipses (shown in blue) is used for a search over pose space (heat map, upper middle). Upper right: the modes of the distribution are shown in red over the original user specification in blue. Bottom: we show the three training images that are closest in the latent manifold space to the mode marked as a purple dot.

specifies how likely the configuration is under the training data and may be used to threshold the results to prevent unlikely ellipse configurations being presented to the user. In addition to the variance threshold, we constrain the returned ellipse parameters configurations such that ellipses that correspond to the existing ellipses sketched by the user are within a certain threshold. More information on the cost and variance is provided for the silhouette query in the following paragraph and in Figure 3.9, but the discussion is equally valid for the ellipse queries.

Contour Manifold Queries Figure 3.8 demonstrates the multi-modal nature of the distribution of the contours with respect to the ellipses. In order to specify a particular silhouette, we allow the user to sketch parts of the contour. Just as we did above for an ellipse query, we define a distance function between the contour of a point on the manifold and user sketches: we use the chamfer distance [225] between the user sketch and the contour under a truncated-quadratic cost function. Since the cost function is truncated, the query results

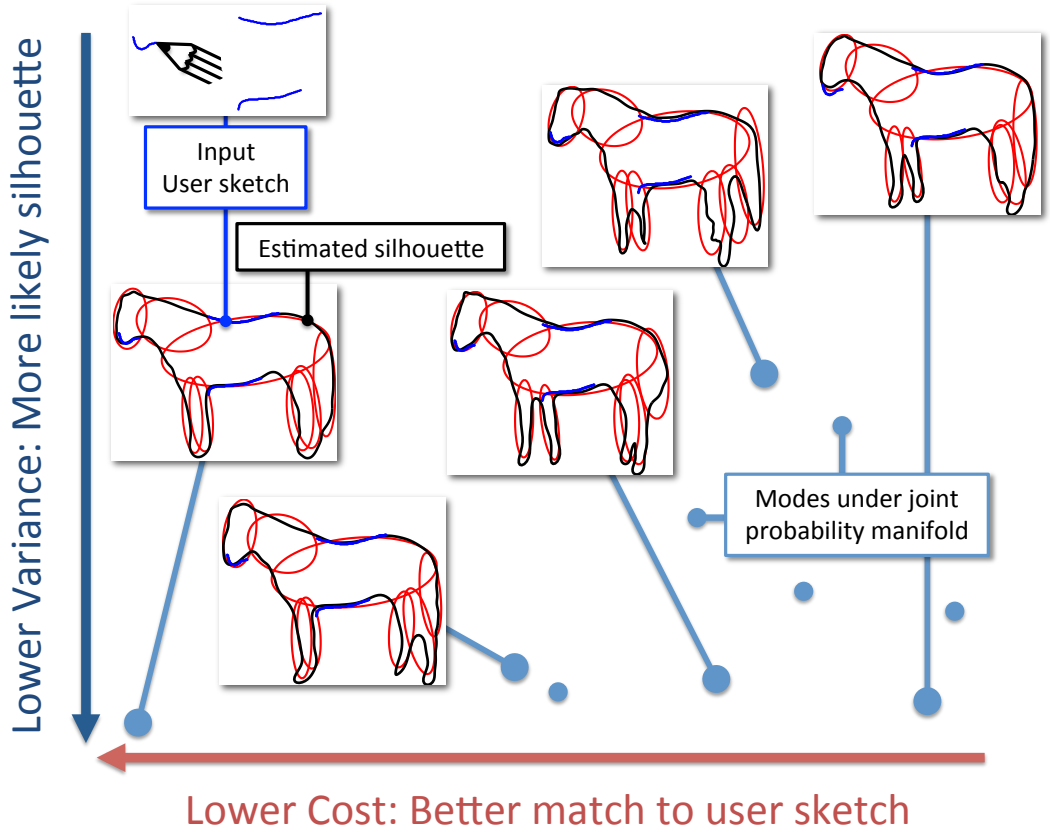


Figure 3.9: The result of a user drawn contour fragment query on the joint manifold for the horse images. The user has drawn three sketches, shown in blue that are used to define the query. The query returns the modes of the likelihood distribution over the joint manifold conditioned on the user query. Each result has a cost and variance associated with it, the two axes used to plot the modes in the figure. The modes with lower costs correspond to silhouette that match the user sketch more closely and the modes with lower variance represent silhouettes that are more representative of the training data and thus should produce better results from image melding. Examples of the estimated silhouettes and ellipses associated with some of the modes are provided.

should be robust against incomplete and outlier sketches that the user may draw.

This defines the similarity between the user sketches and a point on the manifold. We then perform a set of optimizations (as described above for ellipses) to find the modes in the manifold. Since the contour function is more expensive to compute, we accelerate the search by using the modes of the ellipse query as the initializations. These queries typically take around half a second.

3.4.3 Sketching Masses and Contours

In the first stage of sketch interaction, we ask the user to scribble some or all of the masses. The ellipses that represent masses can be manipulated by dragging control points, but in order to make the user interface more intuitive, we also allow freeform drawing of masses, and fit the freeform strokes to a set of parameterized ellipses.

The user can sketch ellipses using one or many strokes, specifying ellipses either partially or completely. To solve this problem, the system assigns each stroke to one ellipse, and each ellipse is fitted to all of its assigned strokes. As the user inputs a new stroke, the system computes the cost for each of the ellipses of the current set of ellipses

$$\text{cost}(i) = \frac{1}{||\{S^\star \cup S_{\epsilon_i}\}||} \sum_{S_k \in \{S^\star \cup S_{\epsilon_i}\}} \sum_{p \in S_k} \text{dist}\left(p, \epsilon_i^\star\right) , \tag{3.2}$$

where $S^\star$ is the new stroke, S_k is the k-th stroke of the user, S_{ϵ_i} is the set of strokes assigned to ellipse i, p is a point of the stroke S_k, $\epsilon_i^\star$ is the ellipse that was fitted to $\{S^\star \cup S_{\epsilon_i}\}$ and $\text{dist}\left(p, \epsilon_i^\star\right)$ is the distance from point p to the ellipse $\epsilon_i^\star$.

This cost computes the average of the distances between the strokes that were assigned to the ellipse and the corresponding fitted ellipse. If the new ellipse stroke does not fit any of the previous fitted ellipses (the average distance is more than 40 pixels for each of the ellipses of the current set of fitted ellipses), the system creates a new ellipse. Otherwise, the new stroke is assigned to the best matching ellipse from the current set of fitted ellipses. This approach also allows the user to erase incorrect strokes and change ellipses by drawing over the top of existing ellipses. As long as the "overdrawn" strokes are nearby, the ellipse will fit all of the assigned strokes.

Having obtained a set of ellipses, we need to identify the corresponding body part label for each ellipse. This is performed automatically in two steps. First, we find the set of ellipses from the images of the dataset that are closest to the user's ellipses, by minimizing

$$i^* = \arg\min_i \sum_k \min_{j \in \mathcal{P}} \sum_{p \in \epsilon_k} \text{dist}\left(p, \epsilon_{i,j}\right) , \tag{3.3}$$

where $\epsilon_{i,j}$ is the ellipse fitted into the body part j of the labeled image i, and $\mathcal{P}$ is the set of body-parts, p is a point of ellipse ϵ_k, ϵ_k is the k-th ellipse fitted

to the user's scribbles, and dist $(p, \epsilon_{i,j})$ is the distance from point p to the ellipse $\epsilon_{i,j}$. As the number of parts in $\mathcal{P}$ is low (~ 10) we can find the solution with an exhaustive search.

We then obtain the part label for each of the user's ellipses

$$j^* = \arg\min_j \sum_{p \in \epsilon_k} \text{dist}\,(p, \epsilon_{i^*,j}) \ . \tag{3.4}$$

Once we assign a label j^* to ellipse ϵ_k, we remove j^* from the set of possible labels to ensure a unique assignment. We allow the user to override the assigned labels if desired. The operations above can be computed efficiently by precomputing chamfer distances [225] for the training images in the dataset and reducing the resolution of the query ellipses.

The proposed ellipse and label assignment proved to be robust and efficient, allowing us to fit ellipses to the user strokes and to infer their labels as the user adds new strokes to further define the pose of the object.

When the user is satisfied with the gross arrangement of masses, she can switch to contour mode. In this mode, successive pen strokes specify the boundary of the final synthesized object. Similarly to drawing the ellipses, multiple partial strokes are supported and previous strokes may be erased.

3.4.4 Appearance Constraints and Synthesis

Using the user's fully or partially specified pose, we can retrieve multiple training images that have similar poses. We do this by finding modes on the manifold that match the ellipses and contours with a low cost and low variance and by taking the nearest neighboring training images, see Figure 3.8. Within this set of neighboring images, we can identify the range of possible appearances available in our dataset, for this specific pose, to use for image synthesis. We present color palettes computed from exemplars of the possible appearances to the user and allow them to select the most appropriate (see right side of Figure 3.1).

The data-driven nature of our algorithm means that we rely on the appearance variation in the training images to produce the appropriate variation during image synthesis.

Features The specified masses, contour strokes and appearance constraints, together with the inferred masses and contours, can be converted into features that guide the synthesis process. In addition to CIE *Lab* color channels, we have a feature channel per ellipse and another computed from the contour. The

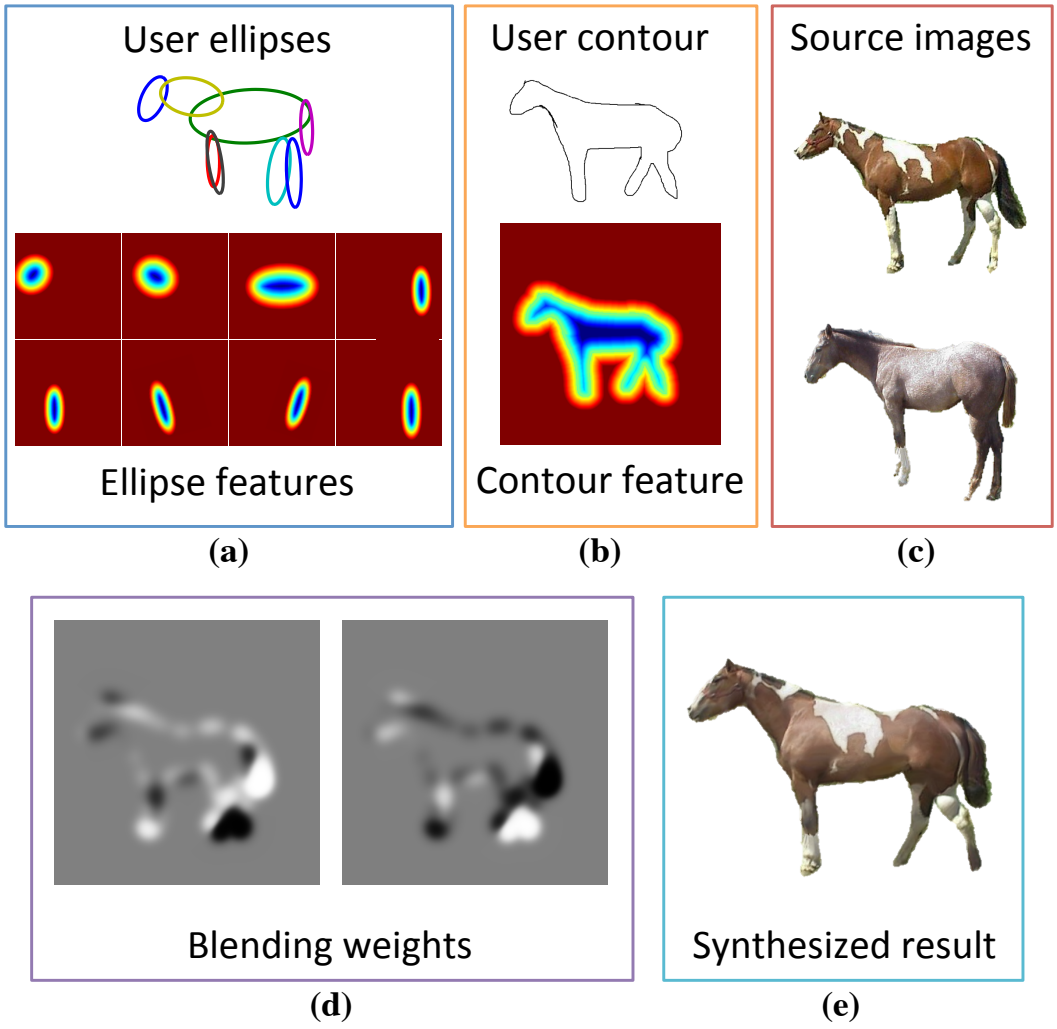


Figure 3.10: Example synthesis result. (a) The ellipse configuration is used to produce a set of features (one channel per ellipse) that are combined with (b) the feature channel from the contour and (c) the CIE *Lab* channels of the nearest neighbor source images as an input to synthesis. (d) The blending weights for each image are computed from blurred distances between the source image feature channels and the target feature channels. (e) The synthesized result.

additional channels allow semantically meaningful synthesis.

Each ellipse feature channel is a truncated signed distance to the nearest point of the boundary of the ellipse, and the contour feature channel is a truncated signed distance to the nearest point of the contour. If the contour provided by the user is not closed, we estimate the most likely contour using the GP-LVM manifold. The feature channels are also computed for the nearest neighbor source images of the dataset and for the target image under the appearance constraints. Figure 3.10(a-b) provides examples of these feature

channels.

Synthesis Synthesis of the target image is done using the image melding framework [159], using the feature channels as guiding layers, as in Image Analogies [171]. For efficiency we use, but are not limited to, the two nearest neighbor images that are closest to the user's specifications on the GP-LVM manifold. To ensure high contrast along the contour, we double the weight of the contour feature with respect to the other features. At the coarsest scale of the image pyramid, we initialize the target image by computing the nearest neighbor patch correspondences using only the feature channels. Subsequent iterations use both feature channels and color channels.

At each successive scale, we compute a correspondence map from each of the source images to the target image. The target image is reconstructed using the patches of the source images according to this map and using the reconstruction costs for each source to compute blend weights; see Figure 3.10(d).

Figure 3.10(e) shows an example synthesis result from two source images. Given the large number of feature channels, the synthesis step takes around 4 minutes to perform at all scales. Thus, during the user interaction, we do not perform the synthesis up to full resolution. Instead, we produce an approximate synthesis at low-resolution, and upsample the resulting nearest neighbor field to full resolution. This enables our method to efficiently synthesize a full resolution "preview" image using high-resolution patches at interactive rates (about 3 seconds for an image). Although this produces some artifacts due to upsampling, the resulting preview is a reasonable proxy for the appearance of the final synthesis.

3.5 Synthesis Results

We have compiled 4 datasets: horses, pigeons, elephants and cats. The horse dataset was compiled using 295 images and segmentations of the Weizmann horse dataset [219, 220, 221]. We also collected images of pigeons (270 images) and elephants (275 images) from the internet and manually segmented the images. We have acquired a single photocollage of 390 cats on the white background captured by professional photographer. All the datasets were hand-labelled into corresponding parts. Since the labeling does not have to be perfect, each image can be labelled in about 4 minutes. Each of the images in the dataset was rescaled, cropped and segmented from the background. The scaling was chosen such that the area of the torso matches in each image.

Figure 3.11 shows some results created with our method along with the

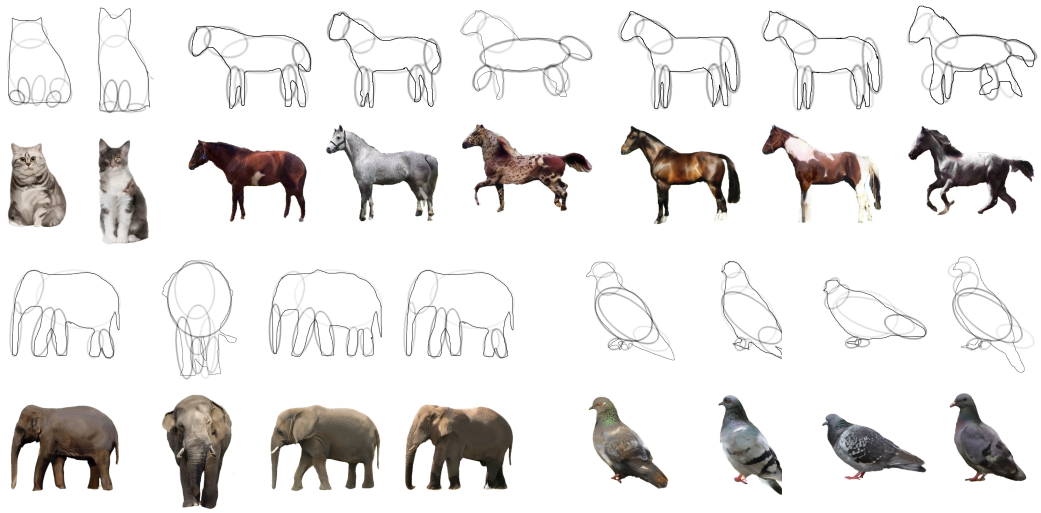


Figure 3.11: Synthesis results for the horses, elephants, cats and pigeons datasets, with sketched masses and contours. Each image is a combination of training images, not simply the most similar image from a database. Note the quality of the results and their agreement with the specified masses and contours, in spite of the relatively small database sizes. At top right, the user ignored the shadow suggestions and drew a contour for the horse that is inconsistent with the masses for the front legs: This results in a synthesis failure due to incompatible constraints. See Appendix C for high-resolution images of the results and images of the nearest neighbor images used for the image synthesis.

user-drawn masses and contours that produced them. Each sketch took only about 2 minutes to draw. The synthesized images appear realistic and follow the user's constraints closely. Notice that the method allows results to be produced over a wide range of poses. Please see the Appendix C for an expanded version of Figure 3.11 containing details of the nearest neighbor images used for the image synthesis.

Our system is not computationally expensive. The preprocess of fitting ellipses and fitting contours to the horses in the database takes 20s and 30s, respectively. Training the GP-LVM takes about 2 minutes on the same dataset. Querying the closest ellipses at run-time is interactive at 0.4s. Querying partial silhouettes takes about 0.5s. Synthesizing a preview result takes 3-4 seconds. The final, high-quality synthesis is more expensive, and usually takes around 3 minutes to compute depending on the resolution of the images.

Visual Feedback	System 1	System 2	System 3
Draw Ellipses	x		
Silhouette Shadow	x	x	x
Fast NN Preview	x	x	
Coarse Preview	x	x	

Table 3.1: Visual Feedbacks corresponding to the Systems.

3.6 User Studies

3.6.1 First User Study

Our method was designed to help users generate images.To assist the user in accomplishing this task, the method displays previews and shadows that work as a guide for the user's input. The shadow is generated both when working with ellipses and contours. Therefore, to evaluate the usefulness of our method we conduct a user study in which participants are asked to generate an image as close as possible to a target image. We evaluate the usefulness of each of the feedback visualizations: the participants perform three assignments by using three variations of our system that each employ a different set of visual feedback. For a subjective assessment of the generated image, we also assign a Manual Search assignment that asks the participant to select an image from the dataset that is closest both in terms of pose and color to the target image. We provide the target image to evaluate the subjective quality of the synthesized result. After completing the assignments, the participants are surveyed with the standard System Usability Scale questionnaire [226] to provide a subjective assessment of general usability, as well as system-specific questions to evaluate each of the visual feedbacks' usefulness, and results of interaction with the system. We chose the horses dataset for the first user study.

3.6.1.1 Assignments

The Manual Search assignment requires the participant to browse through 295 images of horses to find the closest match to a target horse both in terms of pose and appearance. Each of the 295 images was rescaled, cropped and segmented from the background.

In Assignment 1, the participant uses our complete system to generate an image that is as close as possible to the target image (same target image as in the Manual Search assignment). The "System 1", used in Assignment 1, provides all visual feedbacks. Assignment 2 uses "System 2," a limited version of "System 1;" the user is not allowed to use ellipse interactions and starts with the

"Draw Contour" tool. Finally, Assignment 3 uses "System 3," a further restricted system. The user is not allowed to use ellipse interactions and starts with the "Draw Contour" tool. "System 3" does not generate any kind of preview. "System 3" is similar to the Shadow Draw [197] system, but with the addition of interpolated contours and an image synthesis post-process. Our hypothesis is that additional visual feedback aids in the interactive image synthesis task. Table 3.1 shows the supported visual feedbacks for the Assignments.

3.6.1.2 Data Collection and Participant Selection

18 participants from the student population of our department performed the user study. Each participant was randomly assigned 3 different target images for the four assignments (Manual Search and Assignment 1 share the target image) from a set of 6 images. We did not filter the study population for handedness. Only 2 of the participants were familiar with the concept of "masses". To minimize the influence of learning effects, the assignments are conducted consecutively starting with "System 1" featuring the full set of visual feedbacks. The goal of the study was not mentioned to the participants. All participants were familiar with image editing in general and were given training using a Wacom tablet.

3.6.1.3 Procedure

First, we allow the user to familiarize themselves with the Wacom tablet. We allow using the mouse for the experiment, but all of the participants preferred the Wacom tablet. Before starting the tasks, the participants were asked to answer two questions regarding their artistic skills and artistic training. Then, the participants were asked to perform the Manual Search assignment using a randomly assigned target image. Next, the participants were shown a video tutorial describing the system. To clarify the part subdivision and the relationship between ellipses and parts, an example image was given to the participants. All assignments were performed without time limit. Synthesizing the final result was done on a separate machine in the background. After finishing the assignments, the participants answered the SUS questionnaire. Before conducting the system related questionnaire, we recapitulate the differences between systems and show the final rendered results. Finally, we ask the participants to rate the usefulness of each of the visual feedbacks on a Likert scale (see Table 3.2). The questions of the survey and the user responses are listed in the Appendix D.

System related questions	Number of Votes				
	SD	D	NN	A	SA
Ellipse position feedback was useful?	0	5	1	11	1
Silhouette feedback was useful?	0	1	1	7	9
Fast "NN" preview was useful?	0	0	3	9	6
Coarse preview of the generated image with the color choices was useful?	0	0	2	9	7

Table 3.2: First user study: number of votes the visual feedback related questions. The Likert response scale answers are "Strongly Disagree" (SD), "Disagree" (D), "Neither Agree Nor Disagree" (NN), "Agree" (A), "Strongly Agree" (SA).

3.6.1.4 Expectations

We expect the system to score above average on the SUS scale (corresponds to a SUS score above a 68). [227] The system was designed to assist the user through visual feedbacks, hence, we expect that the participants would evaluate all visual feedbacks as "useful". We assume that "System 1" with the full set of visual feedbacks, including the ellipse interactions, would be evaluated as the easiest and the most efficient in accomplishing the assignment, given that it provides the most visual feedback.

3.6.1.5 Results

The average score of the system on the SUS scale was 68.75, which corresponds to "above average" [227]. Unexpectedly, both "System 1" and "System 2" received same amount of votes (9 votes each) as the easiest and the most efficient. One explanation of this result may be in the inherent preference of users to sketch without the use of masses, as we do not *teach* the users to draw using masses, and users inexperienced at drawing may not be familiar with the concept. We only show examples in the tutorial video and before Assignment 1. Moreover, the first and only trial of drawing with the masses in Assignment 1 may not be enough to fully grasp the concept. Some of the participants said that they believed they would've performed better in Assignment 1 if they were to reuse "System 1" after completing the survey. We hypothesize that the data in Table 3.2 supports this point as the utility of the ellipse position feedback appears to be bi-modal with two thirds of the study participants finding the ellipse position feedback useful.

We also asked the users to compare the generated image of "Assignment 1" with their own choice from the dataset and the nearest neighbor found by the

Image comparison questions	Number of Votes		
	Generated Image	Manual Search	System's Choice
Which one is the closest to the target image in terms of pose?	5	10	3
Which one is the closest to the target image in terms of color?	5	12	1
Which one resembles the target image the most?	7	11	0

Table 3.3: First user study: number of votes for the subjective assessment of the synthesized image of the "Assignment 1".

system based on the user's sketch (this ignores the appearance choice). The results can be viewed in Table 3.3. The generated image quality depends on the complexity and uniqueness of the target pose, the quality of the sketch and specifications, the size of the dataset *etc*. Nonetheless, about third of the participants found the generated image more closely resembled the target image than the manually selected image.

3.6.2 Second User Study

In our second user study, participants were asked to create novel images of elephants, using our elephants database containing 275 images. Whereas the horses dataset has only profile views, the elephants dataset has more diverse pose variations in 3D, including both frontal and profile views.

In the first study we provided participants with a target image in order to allow post hoc comparative evaluation of the results, but in practice, real users of our method would not have such a target image. Thus, in second study we do *not* provide participants with a target image, instead, asking users to draw a sketch of an elephant pictured solely in their mind's eye. As in the first user study, the users interact with System 1 and System 2; and afterwards are surveyed with the questionaire. In this user study, we omitted the image comparison questions which are not relevant without a target image, instead, focusing on the usefulness of different components of the method. Furthermore, we omit the SUS questionnaire as the usability of the system has already been evaluated in the first user study.

Table 3.4 shows the votes on the feedback-related questions. All participants preferred System 1, and all participants found ellipses useful for specifying pose.

	Number of Votes				
System related questions	SD	D	NN	A	SA
Ellipse position feedback was useful?	0	0	1	3	2
Silhouette feedback was useful?	0	1	0	3	2
Fast "NN" preview was useful?	1	0	2	2	1
Coarse preview of the generated image with the color choices was useful?	0	0	0	3	3
When specifying pose, ellipses were useful?	0	0	0	4	2

Table 3.4: Second user study: number of votes the visual feedback related questions. The Likert response scale answers are "Strongly Disagree" (SD), "Disagree" (D), "Neither Agree Nor Disagree" (NN), "Agree" (A), "Strongly Agree" (SA).

3.7 Conclusion

We have presented an interactive method for synthesizing realistic objects based on user input, given a database of training images. Our method supports a traditional illustrator's workflow, whereby the user first sketches the important masses and then refines them using contours. The advantages of this approach are the same as in traditional illustration: The gross pose of the figure can be specified loosely and iteratively, without requiring precise or complete contours. Interactive feedback is provided by indicating likely mass locations and contours to the user, as well as quickly synthesizing a preview of the object. This feedback aids novice users in understanding the pose space as they construct their sketch, and visually indicates likely outcomes for the synthesis phase. Although in this chapter we demonstrate the drawing of only a few classes of objects, our general approach can be extended to other classes of objects. Positive feedback from the users indicates a promising pathway to advance interactive tools for creative illustration workflow.

Our system is both straightforward to use and computationally efficient. It produces high-quality results that generalize the training images by analysis of the joint manifold model, and is capable of interpolating a wide range of poses from the sometimes sparse training poses. The user's input also constrains the non-parametric image synthesis algorithm to generate results that are in the manifold of valid and realistic images. The capabilities of image synthesis algorithms are presently outstripping the interactions used to control them. We hope our method inspires further research in interactive control of structured image synthesis.

3.8 Limitations

Our present method is not without its limitations. Firstly, it is tuned for sketching animal figures. We rely on the fact that the structure of animal figures is fixed (the head is attached to the neck, and so forth), which is not true for general objects or scenes.

We also assume that ellipses are a good representation of the shape of the body parts. Although our method gracefully handles cases where this assumption does not hold (see the horses' legs), it would be straightforward to support additional primitives to better represent a wider range of figures.

In order to synthesize realistic figures, we require the user to provide constraints (masses and contours) that are reasonably similar to some poses in the training images. If the user veers too far from the database poses, synthesis results may be unsatisfactory. An example can be seen in the top right of Figure 3.11, in which the pose of the foreleg masses is inconsistent with the specified contour. However, if the user allows themselves to be guided by the visual feedback, the final sketch should reside in a valid location on the manifold with sufficient training images for synthesis. In this work we err on the side of giving the user more creative control at the cost of potentially less plausible results, but it is straightforward to automatically override a user's constraints towards higher-probability locations in the manifold. This tradeoff between flexibility and plausibility warrants further exploration.

We currently require annotated images as input, which is labor-intensive and limits us to a sparse collection of images (on the order of a few hundred). Databases of images with semantic part labelings [228, 100, 229] are becoming more widely available for addressing computer vision tasks, but in the long run, we hope to take advantage of current and future advances in computer vision to automate this part of our method [230, 231].

In our current method, we have not added local texture constraints, as the limited database sizes makes them hard to use.

3.9 Discussion

As it was mentioned, we made a number of assumptions that have caused some of the limitations above. Here we discuss some of the extensions we would desire for our method that would alleviate the assumptions and consequently generalize our method for a wider set of objects or richer ways of interaction.

In the current implementation, our synthesis process creates a hybrid or a

blend of two images that have similar, but not necessarily identical pose. Our motivation for using only a few images is as follows. First, we believe that the chance of the high-frequency signal of the source images being blurred away is reduced, as the lesser the number of images being averaged, the sharper the image would be. Second, choosing a low number of images for synthesis is convenient for the blending of the colors, as an average of multiple colors may be unrealistic (think of an average of a spotted horse and a brown horse). So, ideally we would like to explore integrating the color and texture constraints more directly into the continuous exploration mode of the pose specification. For example, we could have a few continuous parameters that would control the appearance of the object, *i.e.* a few dials, where one changes the horse from black to white and another changes the horse from uniform color to a spotted pattern. This would require a parametric model of the appearance. However, the change of appearance should not change the sketch that was already specified by the user. Furthermore, the space of appearances should not depend on the pose sketched by the user. Hence, the desired parametric model should be independent of the pose.

Another important assumption is the fixed structure of the visual objects. This may be inconvenient to the user, if she is only interested in a partial view of the horse. Ideally, the system should be able to recognize this and use corresponding partial views from the training set. Moreover, there are classes of objects that do not have a fixed structure, but have important correlations in the appearance: flowers with a different number of petals, chairs with a different number of legs, facades with a different number of windows, *etc.*

So, we have identified two major extensions: modeling appearance of an object with a few parameters while the pose is fixed and modeling appearance of an object with a varying structure. In the next chapter we present a model that directly tackles these two extensions.

Acknowledgements The authors would like to thank Eli Shechtman for valuable discussions and Leah Anton for the sketches of horses in Figure 3.2.

Chapter 4

Context-Conditioned Component Analysis

Subspace models have been very successful at modeling the appearance of structured image datasets when the visual objects have been aligned in the images (*e.g.*, faces). Even with extensions that allow for global transformations or dense warps of the image, the set of visual objects whose appearance may be modeled by such methods is limited. They are unable to account for visual objects where occlusion leads to changing visibility of different object parts (without a strict layered structure) and where a one-to-one mapping between parts is not preserved. For example, bunches of bananas contain different numbers of bananas, but each individual banana shares an appearance subspace.

In this chapter we remove the image space alignment limitations of existing subspace models by conditioning the models on a shape dependent context that allows for the complex, non-linear structure of the appearance of the visual object to be captured and shared. This allows us to exploit the advantages of subspace appearance models with highly-deformable objects whilst also dealing with complex occlusions and varying numbers of parts. We demonstrate the effectiveness of our new model with examples of structured inpainting and appearance transfer.

4.1 Introduction

Subspace models are commonly used to learn a parametric model of a visual object appearance from a large dataset of images. The parametric model captures low-dimensional representation of the appearance of the visual object as a linear combination of "components". Such representations have been

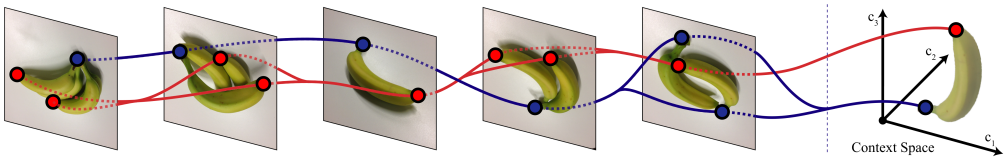


Figure 4.1: An example of an unstructured dataset: due to the multiple instances and occlusions there is no clear way of aligning the images via a global transformation, or indeed estimating a dense warping (one-to-one mapping) between them. However, additional information, such as part labeling or segmentation of the bananas, can be used to *align the data in the context space* (right hand side of the figure) and allow us to learn a subspace model of the appearance of the bananas. In this example, the red and blue dots denote corresponding locations in the context space.

successfully used to model the appearance of the visual object in a range of computer vision and graphics applications, such as face detection, identification, image hallucination, image synthesis, *etc*.

These parametric models yield good results when images are preprocessed so that features of the visual objects are *aligned* throughout the dataset. This is applicable to visual objects that have fixed structure and do not have significant deformations or occlusions, for example frontal images of faces, medical scans of organs, *etc*. Some works address a more challenging case of visual objects that exhibit deformations by jointly learning spatial transformations and appearance representations (*e.g.*, unaligned images of faces, images of mushrooms, medical scans of hands). The models rely on estimating dense mappings between the images of the dataset, usually to a common template.

However, visual objects with varying structure (Figure 4.1) cannot be readily expressed using previous methods. By fixed structure, we mean that images consist of a *fixed set* of regions that have certain associated textures. For example, side views of cars, in general, have *two* regions corresponding to two wheels and these regions are always present in the image. On the other hand, an example of a visual object with a varying structure would be a facade of a building with a *varying number* of windows. In this scenario, the texture of window regions can be shared across images, but each image may have a different number of regions corresponding to windows. Furthermore, datasets of visual objects that have occlusions or significant deformations which can cause self-occlusions or 2D topology changes also remain a challenging input, *e.g.*, animals under different poses. We note that, due to the diverse structure of such images, methods that estimate a warping or dense mapping between images,

such as SIFT flow [67], are unlikely to succeed as some of their assumptions about the images are not met; *e.g.*, smoothness of the deformation field and the presence of occlusions.

Recently, image datasets of various objects have been provided with high-quality segmentation maps and per-pixel semantic and part labels; such labeled datasets are becoming more prevalent [74] and are being used to tackle a variety of visual challenges. We exploit such additional information by formulating a generative parametric statistical model that explains a visual object's appearance *conditioned* on the additional information, such as the segmentation map, but in a more complex way than just warping. So, *we assume that the additional information of each image is known* during training, as well as during sampling.

Existing methods for statistical shape modeling, for example the recent ShapeBM model [139], produce generative models of shape that could be adapted to include part labels. Alternatively, *discriminative* approaches that learn part labels, for example [232], can be used to produce part segmentations to train our *generative* model of appearance.

Our proposed subspace model allows us to model the appearance of objects with varying structure, such as animals and facades of buildings. We demonstrate the performance of the proposed subspace model on the task of structured inpainting of unobserved test images, and by transferring of the appearance of one instance (*e.g.*, in a specific pose) to others (*e.g.*, different poses).

4.2 Related Work

Aligned Subspace Appearance Models One of the earliest works in learning a statistical model of a visual object from a dataset of images was Eigenfaces [16], by Turk and Pentland, that was used to address the problem of face detection and identification. Their approach expressed each image of a face as a feature vector of pixel intensities. These feature vectors are approximated as a linear combination of a subset of principal components derived from the covariance matrix of the probability distribution over the vector space of the face image pixel intensities.

Many related approaches followed; for example, the Fisherfaces algorithm [26] makes use of linear discriminant analysis (LDA). In this work, the subspace aims to maximize the ratio of inter-individual to intra-individual variance. The null-space LDA approach [27] makes use of the directions without

intra-individual variance (these are ignored by Fisherfaces). The Dual-Space LDA approach [28] combined both of these subspace directions. All of these approaches rely on the image datasets to be aligned.

Active Appearance Models The Active Appearance Model (AAM) [8] by Cootes *et al.* is a more sophisticated statistical model that models both the shape and the appearance of a visual object from a dataset that consists of images and corresponding coordinates of a set of landmark points on each image. AAM extends subspace models by incorporating parameterized affine warps of all images to a common template. Edwards *et al.* [43] used AAM to address the problem of face detection and identification. The AAM can be applied to image alignment problems that estimate transformations of images to a common coordinate system or a common template [233].

Jones and Soatto [9] proposed a Layered Active Appearance Model which allows for missing features, occlusion, substantial spatial rearrangement of features, and that provides a more general representation that extends the applicability of the traditional AAM. In LAAM the images are subdivided into "layers" (a set of compact regions determined by a pre-defined group of the landmark points with pixel's local coordinates, pixel intensities and landmark point coordinates) which are modeled with weighted PCA. Jones and Soatto demonstrated the model on a dataset of cars. The dataset consisted of frontal images of cars with and without certain features (such as foglights) and overlapping features (such as license plates). However, LAAM requires the ordering of layers to be fixed, *i.e.*, foglights always occlude bumpers and bumpers never occlude foglights. Such an ordering assumption is not always applicable (*e.g.*, consider the legs of animals). Also, the number of instances of each part must be known beforehand (a fixed number of layers), which is not required for our context-based approach. The Morphable Model [49] extended the AAM into 3D for modeling faces which also helps with occlusions; however, it still cannot handle varying numbers of parts and requires alignment to a template.

Combining Parametric and Non-Parametric Models for Appearance Synthesis Liu *et al.* combined a parametric subspace model and a local non-parameteric model in a two-step procedure to solve the problem of face hallucination [205, 204]. At the first step, principal components are learned from the dataset to express the relationship between the high-resolution face images and the corresponding blurred and downsampled lower resolution images. At the second step, the residue between an original high-resolution image and

the image reconstructed by using the learned subspace model is modeled by a patch-based non-parametric Markov network to hallucinate the high-frequency details of the image.

Visio-lization [1] by Mohammed *et al.* used a similar framework to address the problem of synthesizing frontal face images. First, a parametric subspace model is sampled to generate a low-resolution novel image which lacks high-frequency details. The second step upsamples the low-resolution image by performing image quilting [155] on overlapping image regions. Image quilting uses patches extracted from the images of the dataset that were conditioned on the low-resolution image.

Jointly Estimating Deformation and Appearance Transformed Component Analysis [69] estimates a global transformation for each image in the dataset to bring the images into alignment as well as finding an appearance subspace. The set of transformations they consider are rigid body motions and thus not suited to highly-deformable objects or occlusions.

Winn and Jojic introduced LOCUS [70], an unsupervised generative model that learns segmentations of visual objects. Most interestingly, they model visual objects with deformation fields and achieve dense registration between different images of the same class despite differences in appearance or pose.

Mobahi *et al.* [71] also learn a model of a visual object from a set of images. They assume that images are generated as a nested composition of color, appearance and shape transforms. By modeling each component as a low-dimensional subspace they can learn a regularized solution of such compositional model from a set of images. Their shape(geometric) transform is jointly learned for all images, and outperforms SIFT flow and robust optical flow for any pair of images from the set.

Both the LOCUS model and the work of Mobahi *et al.* make use of a deformation (or flow) field for dense warping. Such a field cannot encompass the range of transformations that we address with our approach (complex occlusions and varying numbers of instances).

4.3 Context-Conditioned Component Analysis

We begin by providing an illustrative example that reveals the limitations of existing approaches; we use this example to motivate the use of our method for modeling the appearance of complex visual objects. We then provide a high-level description of our new model that we call Context-Conditioned Component Analysis (C-CCA). We show a general formulation and demonstrate that our

model encompasses Probabilistic Principal Component Analysis [5] (PPCA) as a special case of the proposed model in section 4.7.1.

For simplicity, we will begin by assuming that the images are greyscale, but we relax this to include the case of color images in section 4.5.

4.3.1 Motivating Example

Consider images of bunches of bananas as shown in Figure 4.1. This example dataset does not display fixed structure, as every image has a different number of bananas that may occlude one another with no consistent ordering (*c.f.* the layer structure limitation of LAAM). Clearly, aligning these images, or estimating a dense warping between them, is not possible; *i.e.*, deforming 3 bananas to 1 or vice versa. However, the appearance of the bananas in each image is correlated. We argue that additional information, such as part labeling or segmentation of the bananas, can be used to *align the data in the context space* and allow us to learn the subspace model of the appearance of the bananas. In this example, the red and blue dots denote corresponding locations in the context space (*not* fiducial points as used for AAM).

4.3.2 Model Description

Consider a set of greyscale images $\{\mathbf{x}_i\}_{i=1}^{I}$ of a given visual class (e.g. horses, facades, cats). Each image is represented as a vector $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots x_{iJ}]^T$, where the element x_{ij} represents the j^{th} foreground pixel from the i^{th} image. (For simplicity of exposition we assume that all pixels in each image are foreground, however, in practice each image may contain different number of foreground pixels.) Associated with each pixel in each image is a *context vector* $\mathbf{c}_{ij}$. This vector provides some information about the state of the object at that pixel. For example, it might encode the part of the object at that pixel (door, wall of a house), the local shape of the silhouette of the object, or the distance from some pre-determined keypoint in that image. The context vector constrains the intensity values that might be present at a given pixel, but does not entirely determine it.

The model for the j^{th} pixel of the i^{th} image is

$$x_{ij} = \mu\left[\mathbf{c}_{ij}, \boldsymbol{\theta}_\mu\right] + \sum_{f=1}^{F} \phi\left[\mathbf{c}_{ij}, \boldsymbol{\theta}_f\right] h_{if} + \epsilon_{ij} \ , \tag{4.1}$$

where the first term gives the mean value μ_{ij} at the pixel and is a function of the context $\mathbf{c}_{ij}$ at that pixel and a parameter vector $\boldsymbol{\theta}_\mu$. The second term consists

of a weighted sum of F function terms $\phi[\cdot,\cdot]$, where the weights $\{h_{if}\}_{f=1}^{F}$ are constant for the whole image and can be considered hidden variables. Each of the function terms maps the pixel context $\mathbf{c}_{ij}$ to a scalar value, where the F mappings in the weighted sum are determined by associated parameter variables $\{\boldsymbol{\theta}_f\}_{f=1}^{F}$. Finally, each term ϵ_{ij} is an independent stochastic noise variable distributed as $\text{Norm}_{\epsilon_{ij}}[0,\sigma^2]$.

For now, we remain agnostic about the exact form of the functions $\mu\left[\mathbf{c}_{ij},\boldsymbol{\theta}_\mu\right]$ and $\phi\left[\mathbf{c}_{ij},\boldsymbol{\theta}_f\right]$ (and the corresponding function parameters $\boldsymbol{\theta}_\mu$ and $\boldsymbol{\theta}_f$), except to say that they map from the context vector $\mathbf{c}_{ij}$ to a scalar value, and so, take the form of regression functions. We discuss the form of the functions later in section 4.4.5 and the corresponding function parameters in section 4.6.2

We can equivalently write equation 4.1 in vector form as

$$\mathbf{x}_i = \boldsymbol{\mu}_i + \boldsymbol{\Phi}_i\mathbf{h}_i + \boldsymbol{\epsilon}_i \ , \tag{4.2}$$

where the j^{th} row of the $J\times 1$ vector $\boldsymbol{\mu}_i$ is given by $\mu\left[\mathbf{c}_{ij},\boldsymbol{\theta}_\mu\right]$ and the j^{th} row and f^{th} column of the $J\times F$ basis matrix $\boldsymbol{\Phi}_i$ is given by the function $\phi\left[\mathbf{c}_{ij},\boldsymbol{\theta}_f\right]$. We have similarly vectorized the hidden variables so that $\mathbf{h}_i=[h_{i1},h_{i2},\ldots h_{iF}]^T$ and $\boldsymbol{\epsilon}_i=[\epsilon_{i1},\epsilon_{i2}\ldots\epsilon_{iJ}]^T$.

Equation 4.2 can be written in probabilistic form as

$$Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta}_\bullet,\sigma^2) = \text{Norm}_{\mathbf{x}_i}\left[\boldsymbol{\mu}_i+\boldsymbol{\Phi}_i\mathbf{h}_i,\sigma^2\mathbf{I}\right] \ , \tag{4.3}$$

where the notation $\boldsymbol{\theta}_\bullet$ is shorthand[1] for all of the unknown function parameters $\boldsymbol{\theta}_\mu$ and $\{\boldsymbol{\theta}_f\}_{f=1}^{F}$ and $\mathbf{I}$ is the identity matrix. To complete the model, we also define an independent standard Gaussian prior over this hidden variable vector $\mathbf{h}_i$, so,

$$Pr(\mathbf{h}_i) = \text{Norm}_{\mathbf{h}_i}[\mathbf{0},\mathbf{I}] \ . \tag{4.4}$$

Next, we will derive an algorithm for fitting the parameters of the model to the training data.

4.4 Learning

In this section we will discuss how we fit our new appearance model to the training image data. We start by outlining the general learning procedure before discussing the specific approach that we choose. The adopted approach

[1] Throughout we use the symbol $\bullet$ to denote the unravel and concatenate operation similar to the `Matlab` colon operator '`:`', *e.g.*, `Vector = Matrix(:);`.

requires estimating three unknown variables that we discuss in turn in sections 4.4.2 to 4.4.4. For tractability, we choose a special form of the functions $\phi[\cdot,\cdot]$ in section 4.4.5.

4.4.1 Learning Approach

We now outline a general learning procedure. We aim to take a set of images $\{\mathbf{x}_i\}_{i=1}^{I}$ with known context vectors $\{\mathbf{c}_{ij}\}_{i,j=1}^{I,J}$ and estimate the unknown parameters $\boldsymbol{\theta}_\bullet$ and the noise term σ^2 using a maximum likelihood formulation

$$\hat{\boldsymbol{\theta}}_\bullet, \hat{\sigma}^2 = \underset{\boldsymbol{\theta}_\bullet,\sigma^2}{\operatorname{argmax}} \; \log\left[Pr(\mathbf{x}_i|\boldsymbol{\theta}_\bullet,\sigma^2)\right] \; , \tag{4.5}$$

where the log likelihood in this equation is obtained by marginalising out the hidden variables so that

$$Pr(\mathbf{x}_i|\boldsymbol{\theta}_\bullet,\sigma^2) = \int Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta}_\bullet,\sigma^2)Pr(\mathbf{h}_i)d\mathbf{h}_i \; . \tag{4.6}$$

Alternative approaches to learning these parameters include:

1. Integrating out the hidden variables $\mathbf{h}_i$ in closed form and maximize the criteria in equation 4.5.

2. Using the EM algorithm to alternately (i) estimate a probability distribution over the hidden variables to determine a bound on the likelihood (E-Step) and then (ii) optimize this bound with respect to the parameters (M-Step).

3. We could replace the integration in equation 4.6 with a maximization over the hidden variables and alternately estimate the hidden variables and the unknown parameters. This can be shown to be a special case of the generalized EM algorithm.

In this chapter we adopt the third of these three options for simplicity.

4.4.2 Estimation of hidden variables

For the fixed parameters $\boldsymbol{\theta}_\bullet, \sigma^2$ we estimate each of the I hidden variables $\mathbf{h}_i$ as

$$\hat{\mathbf{h}}_i = \underset{\mathbf{h}_i}{\operatorname{argmax}} \; \log\left[Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta}_\bullet,\sigma^2)Pr(\mathbf{h}_i)\right] \; . \tag{4.7}$$

This maximization has a known closed form solution, which can be found by noting that the likelihood term $Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta}_\bullet,\sigma^2)$is a normal distribution with a

mean that is a linear function of $\mathbf{h}_i$ (see equation 4.3) and can be re-written as a constant with respect to $\mathbf{h}_i$ times a normal distribution in $\mathbf{h}_i$. What remains is a product of two Gaussians both in $\mathbf{h}_i$. The product of two Gaussians is also Gaussian and the optimal parameter value will be at the mean of this new distribution. So,

$$\hat{\mathbf{h}}_i = (\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i + \sigma^2\mathbf{I})^{-1}\boldsymbol{\Phi}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i) \; . \tag{4.8}$$

More details can be found in section 7.6.2 of [234].

4.4.3 Estimation of noise

The noise parameter can also be estimated in closed form and is determined by the difference between the model predictions and the observed data

$$\begin{aligned}\hat{\sigma}^2 &= \underset{\sigma^2}{\text{argmax}} \sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}_\bullet, \sigma^2)Pr(\mathbf{h}_i)\right] \\ &= \frac{1}{IJ}\sum_{i=1}^{I}(\mathbf{x}_i - \boldsymbol{\mu}_i - \boldsymbol{\Phi}_i\mathbf{h}_i)^T(\mathbf{x}_i - \boldsymbol{\mu}_i - \boldsymbol{\Phi}_i\mathbf{h}_i) \; . \end{aligned} \tag{4.9}$$

4.4.4 Estimation of function parameters

For fixed hidden variables $\mathbf{h}_i$ and noise σ^2 we estimate the function parameters using maximum likelihood

$$\hat{\boldsymbol{\theta}}_\bullet = \underset{\boldsymbol{\theta}_\bullet}{\text{argmax}} \sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}_\bullet, \sigma^2)Pr(\mathbf{h}_i)\right] \tag{4.10}$$

$$= \underset{\boldsymbol{\theta}_\bullet}{\text{argmax}} \sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}_\bullet, \sigma^2)\right] \; , \tag{4.11}$$

where we have dropped the prior term which does not depend on the unknowns. From equations 4.1 and 4.3, we can write the likelihood from equation 4.10 as

$$Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}_\bullet, \sigma^2) = \prod_{j=1}^{J} \text{Norm}_{x_{ij}}\left[\mu\left[\mathbf{c}_{ij}, \boldsymbol{\theta}_\mu\right] + \sum_{f=1}^{F}\phi\left[\mathbf{c}_{ij}, \boldsymbol{\theta}_f\right]h_{if}, \sigma^2\right] .$$

We can simplify this expression if we assume that the function $\mu\left[\cdot,\cdot\right]$ takes

the same form as $\phi[\cdot,\cdot]$ and then define $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_\mu$ and $h_{i0} = 1$ so that

$$Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}_\bullet, \sigma^2) = \prod_{j=1}^{J} \text{Norm}_{x_{ij}}\left[\sum_{f=0}^{F} \phi[\mathbf{c}_{ij}, \boldsymbol{\theta}_f] h_{if}, \sigma^2\right]. \tag{4.12}$$

When we substitute equation 4.12 into equation 4.10, we see that we are optimising the logarithm of normal distributions and it follows that the parameter estimation takes the form of a non-linear least squares problem

$$\hat{\boldsymbol{\theta}}_\bullet = \underset{\boldsymbol{\theta}_\bullet}{\text{argmin}} \sum_{ij} \left(x_{ij} - \sum_{f=0}^{F} \phi[\mathbf{c}_{ij}, \boldsymbol{\theta}_f] h_{if}\right)^2. \tag{4.13}$$

Solving for a general form of the function $\phi[\cdot,\cdot]$ is not easy, and we would have to apply an iterative method, such as Gauss-Newton, to find a local minimum. Furthermore, a non-linear choice of $\phi[\cdot,\cdot]$ may have multiple local minima solutions.

4.4.5 Choosing the form of the functions $\phi[\cdot,\cdot]$

Fortunately, there is an important class of functions $\phi[\cdot,\cdot]$ for which we can compute a closed-form solution for equation 4.13. In particular, we assume that the functions take the form of a pre-determined non-linear vector function $\mathbf{a}[\mathbf{c}_{ij}]$ of the context vectors; this is then linearly projected onto the parameter vector $\boldsymbol{\theta}_f$, such that

$$\phi[\mathbf{c}_{ij}, \boldsymbol{\theta}_f] = \mathbf{a}[\mathbf{c}_{ij}]^T \boldsymbol{\theta}_f. \tag{4.14}$$

This assumption still allows us to approximate complex functions, since many regression approaches (including Gaussian processes [235], K-Nearest Neighbors [236] and relevance vector machines [237]) can be written in this form. When we substitute equation 4.14 into the least squares criterion (equation 4.13), we see that the solution for the parameters $\boldsymbol{\theta}_\bullet$ now becomes the *linear* least squares problem

$$\hat{\boldsymbol{\theta}}_\bullet = \underset{\boldsymbol{\theta}_\bullet}{\text{argmin}} \sum_{ij} \left(x_{ij} - \sum_{f=0}^{F} \mathbf{a}[\mathbf{c}_{ij}]^T \boldsymbol{\theta}_f h_{if}\right)^2, \tag{4.15}$$

that can be solved in a closed form.

Unfortunately, even this might be difficult to compute in practice. Let M be the dimensionality of the non-linear projection vector $\mathbf{a}[\mathbf{c}_{ij}]$. Then, we are solving a system of (IJ) equations (where I is the image number and J is the

number of pixels per image) and (FM) unknowns, and this may be very slow and memory intensive.

To make this system practical, we use non-linear projections which are sparse (i.e., most of the elements of $\mathbf{a}[\mathbf{c}_{ij}]$ are zero). A simple example of this is *K-Nearest Neighbors* regression [236]. In this case, we compare the context $\mathbf{c}_{ij}$ to M prototype context vectors $\{\mathbf{z}_m\}_{m=1}^{M}$ and $\mathbf{a}$ becomes an $M \times 1$ vector which is zeros except for the K indices corresponding to the K nearest neighbors. These K indices are set to the inverse of the euclidean distance in the context vector space and $\mathbf{a}$ is normalized so that elements sum to 1. We normalize the context vectors to zero mean and unit variance.

4.5 Modeling Color Images

In this section we explain how we extend our model to the case of multichannel color images. The algorithm listing 1 describes how we initialize the variables and summarizes the learning procedure that we use to fit our model to a collection of training images.

To model the appearance, we use our C-CCA on 3 color channels

$$\{\mathbf{x}_i^{(r)}, \mathbf{x}_i^{(g)}, \mathbf{x}_i^{(b)}\}_{i=1}^{I} , \tag{4.16}$$

by using a separate function parameter $\boldsymbol{\theta}_{\bullet}^{(r)}, \boldsymbol{\theta}_{\bullet}^{(g)}, \boldsymbol{\theta}_{\bullet}^{(b)}$ for each of the color channels. We process the colorspace of each image in the training set $\{\tilde{\mathbf{x}}_i^{(r)}, \tilde{\mathbf{x}}_i^{(g)}, \tilde{\mathbf{x}}_i^{(b)}\}_{i=1}^{I}$ with the photometric transformation technique introduced in [71]. This photometric transformation allows us to match colorspaces of different images by modeling the variations of the color of the illumination. Furthermore, the main color of the objects is also modeled by this transformation, which allows the functions $\phi[\cdot,\cdot]$ to learn correlations that are common to visual objects with different colors. For example, the shading of a brown horse is similar to the shading of a white horse, so, this shading variation can be captured by the components and the main color of the horse can be captured by the photometric transformation.

Thus, we estimate a rotation matrix $\mathbf{R}_i$ and translation vector $\mathbf{t}_i$ for each image, that transforms the "common" colorspace modeled by the C-CCA to each of the individual image colorspaces, such that

$$\tilde{\mathbf{x}}_{ij}^{\star} = \mathbf{R}_i \mathbf{x}_{ij}^{\star} + \mathbf{t}_i , \tag{4.17}$$

Algorithm 1 C-CCA Learning Procedure

Require: *RGB values* $\{\tilde{\mathbf{x}}_i^\star\}$, *Context vectors* $\{\mathbf{c}_{ij}\}$
  $\forall i : \mathbf{h}_i \leftarrow$ Random sample of $\text{Norm}_{\mathbf{h}_i}[0, \mathbf{I}]$
  $\forall i : \mathbf{R}_i \leftarrow \mathbf{I}$; $\mathbf{t}_i \leftarrow \vec{\mathbf{0}}$
  $\sigma^2 \leftarrow 1$
  $\forall i, j : \mathbf{c}_{ij} \leftarrow (\mathbf{c}_{ij} - \texttt{mean}[\mathbf{c}]) \,/\, \texttt{std}[\mathbf{c}]$
  $\{\mathbf{z}_m\}_{m=1}^M \leftarrow$ M random samples of $\{\mathbf{c}_{ij}\}$
  $\forall i, j : \mathbf{a}[\mathbf{c}_{ij}] \leftarrow \texttt{kNN}\,[\mathbf{c}_{ij}, \{\mathbf{z}_m\}]$
  for number of iterations **do**
    $\forall i, j : \mathbf{x}_{ij}^\star \leftarrow \mathbf{R}_i^{-1}(\tilde{\mathbf{x}}_{ij}^\star - \mathbf{t}_i)$
    $\boldsymbol{\theta} \leftarrow$ Solution of equation 4.15
    $\sigma^2 \leftarrow$ Solution of equation 4.9
    $\forall i : \mathbf{h}_i \leftarrow$ Solution of equation 4.8
    $\forall i : \mathbf{R}_i, \mathbf{t}_i \leftarrow$ Solution of equation 4.19
  end for
  return $\{\mathbf{z}_m\}, \boldsymbol{\theta}, \{\mathbf{h}_i\}, \sigma^2, \{\mathbf{R}_i\}, \{\mathbf{t}_i\}$

where $\tilde{\mathbf{x}}_{ij}^\star = \left(\tilde{x}_{ij}^{(r)}, \tilde{x}_{ij}^{(g)}, \tilde{x}_{ij}^{(b)}\right)^T$ and the notation $(\cdot^\star)$ is used to denote variables that represent values over 3 color channels.

Since $\mathbf{R}_i$ is a rotation matrix, its inverse can be applied to the images before each iteration of the generalized EM as

$$\mathbf{x}_{ij}^\star = \mathbf{R}_i^{-1}\left(\tilde{\mathbf{x}}_{ij}^\star - \mathbf{t}_i\right) \ . \tag{4.18}$$

The rotation and translation matrices are estimated at the end of each of the iterations by solving

$$\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^I = \underset{\mathbf{R}_i, \mathbf{t}_i}{\text{argmin}} \sum_j \left(\tilde{\mathbf{x}}_{ij}^\star - \mathbf{R}_i \mathbf{y}_{ij}^\star - \mathbf{t}_i\right)^2 \ , \tag{4.19}$$

where $\mathbf{R}_i$ is constrained to be a rotation matrix and $\mathbf{y}_{ij}^\star = \left(y_{ij}^{(r)}, y_{ij}^{(g)}, y_{ij}^{(b)}\right)^T$ is the reconstruction before the color transform with

$$y_{ij}^{(\kappa)} = \sum_{f=0}^{F} \phi\left[\mathbf{c}_{ij}\,,\, \boldsymbol{\theta}_f^{(\kappa)}\right] h_{if} \ , \tag{4.20}$$

for $\kappa \in \{r, g, b\}$. Equation 4.19 has a known closed form solution [238, 71].

4.6 Experiments

4.6.1 Datasets and Context Vectors

Horses We use the Weizmann Horse Dataset [219] that consists of images of horses and corresponding segmentation masks. Additionally, we labeled images with 7 semantic parts (head, neck, torso, each of the 4 legs). The images were rescaled such that the area of the torso matches in each image and translated such that the center of the torso matches in each image. Each pixel's context vector consists of spatial coordinates of the pixel (2 values), and filter responses of the segmentation mask and the 7 part masks. The filterbank consists of 15 filters (a subset of Leung-Malik Filter Bank [239]), namely, the first and second derivatives of Gaussians in 6 orientations, Laplacian of Gaussian at 2 scales and 1 Gaussian filter.

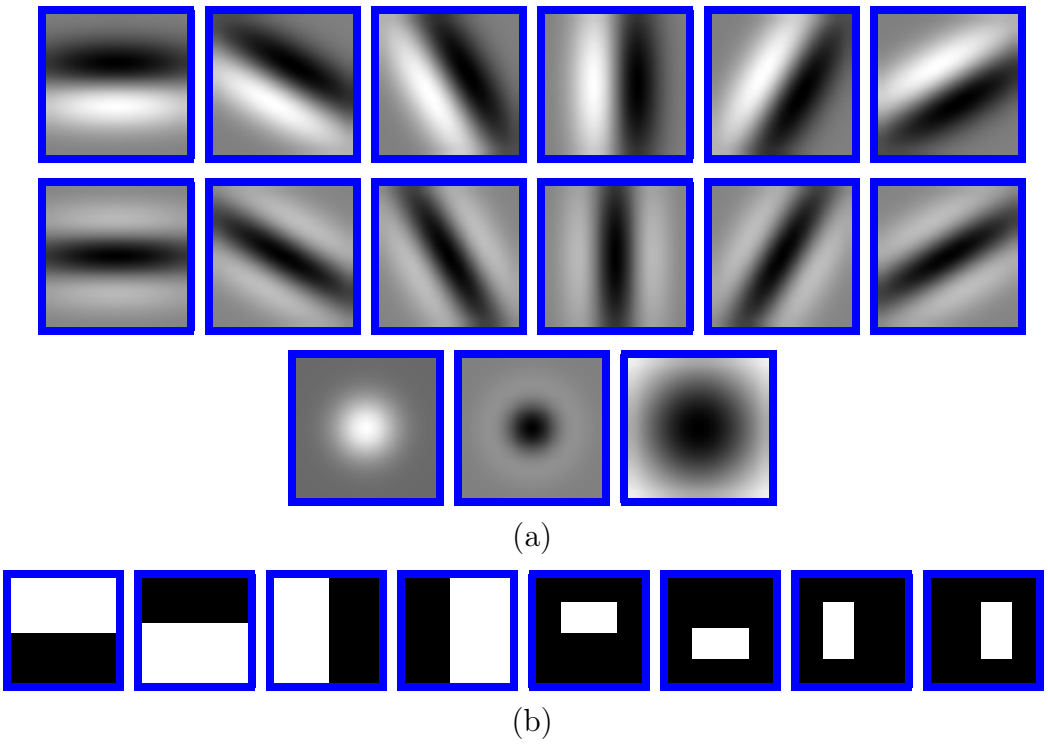


Figure 4.2: Visualization of the filterbanks. The blue color indicates the border of the filters, the black indicates negative numbers and the white indicates positive numbres. (a) The filterbank consists of 15 filters (a subset of Leung-Malik Filter Bank [239]), namely, the first and second derivatives of Gaussians in 6 orientations, Laplacian of Gaussian at 2 scales and 1 Gaussian filter. This filterbank is used on horses, cats and elephants datasets. (b) The filterbank consists of 8 Haar-like features at 2 different scales. This filterbank is used on facades dataset.

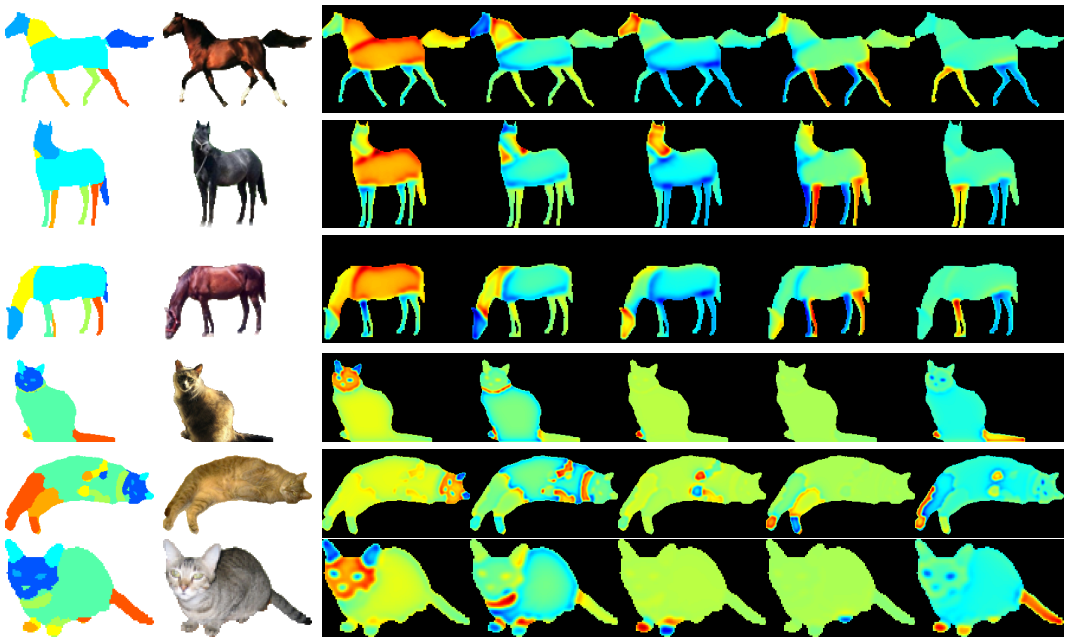


Figure 4.3: Visualization of the context vectors. (a) Structure map of semantic part labels. (b) RGB Image. (c) Projections onto the first five principal components of the context vectors (the true dimensionality of the context vectors is 122 and 271 for horses and cats respectively).

Cats We use images with cats from Pascal VOC2010 dataset [74] with segmentation masks and semantic labels (17 body parts of the cat), provided by [228]. We discarded images that had low-resolution and rescaled images such that the distance between eyes is consistent. Each pixel's context vector consists of the distance to the middle of the eyes and filter responses of the segmentation mask and the 17 part masks. The filterbank is same as for the horses example.

Elephants We compiled a dataset of web images of elephants with corresponding segmentation masks and body part labels (head, torso, trunk and 4 legs). The images were rescaled such that the area of the torso matches in each image and translated such that the center of the torso matches in each image. Each pixel's context vector consists of the spatial coordinates of the pixel (2 values), and filter responses of the segmentation mask and the 7 part masks. The filterbank is same as for the horses example.

Facades We use facades of buildings of Paris from the Ecole Centrale Paris Facades Database [240]. We use pixels that were labeled as wall, window, door, roof or balcony. The images were rescaled such that the average area of the

Dataset	# Images	# Parts	Length of $\mathbf{c}_{ij}$	Average # pixels	Resolution
Horses	295	7	122	3068	150x150
Cats	567	17	271	5069	246x249
Elephants	275	7	122	3672	136x136
Facades	104	5	42	10004	187x174

Table 4.1: Datasets Statistics.

				C-CCA		PPCA	
Dataset	I	Test I	F	K	M	K	M
Horses	200	95	24	16	2000	1	11459
Cats	450	117	16	16	4500	1	44015
Elephants	200	75	20	16	3000	1	11803
Facades	80	24	20	16	2000	1	23629

Table 4.2: Parameters for C-CCA and PPCA.

windows matches in each image. Each pixel's context vector consists of the spatial coordinates and filter responses semantic labeling masks. The filterbank consists of 8 Haar-like features at 2 different scales.

Table 4.1 summarizes the relevant statistics of each of the datasets. Figure 4.2 visualizes the filters used to create context vectors. Figure 4.3 visualizes the context vectors for the horses and the cats datasets.

4.6.2 Quantitative Evaluation

Table 4.2 shows the parameters that were empirically chosen for fitting each of the datasets. Only foreground pixels are used for training and testing. We used the fitted models to compute the reconstruction images of the training and test sets. Table 4.3 shows the per-pixel error of both training and test sets.

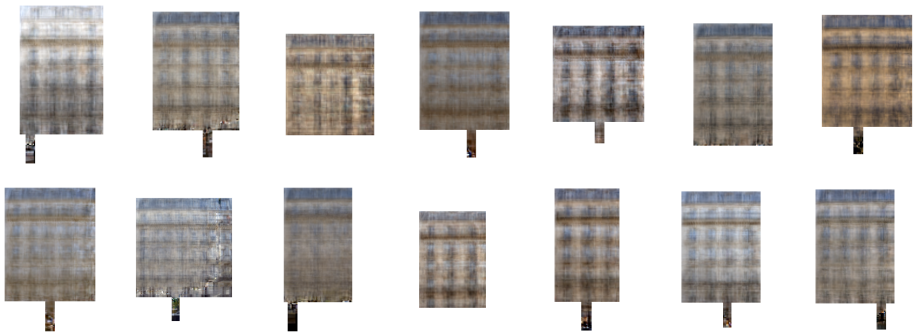
We can compare our model to PPCA, as it is a special case of our model (see section 4.7.1). We change the context vectors to only include pixel coordinates, set K of the k-NN regression learner to 1; and use all of the unique pixel coordinates in our training data as the prototype context vectors $\{\mathbf{z}_m\}_{m=1}^{M}$. We use the same number of components (F) as used for C-CCA for each dataset. The reconstruction error is reported in Table 4.3. C-CCA outperforms PPCA across all datasets while using a significantly smaller number of parameters (M). Examples of reconstructions with PPCA are shown in Figure 4.4.

To analyze sensitivity of the model to the hyperparameters, we fitted

(a) Reconstructions of Horses with PPCA

(b) Reconstructions of Cats with PPCA

(c) Reconstructions of Elephants with PPCA



(d) Reconstructions of Facades with PPCA

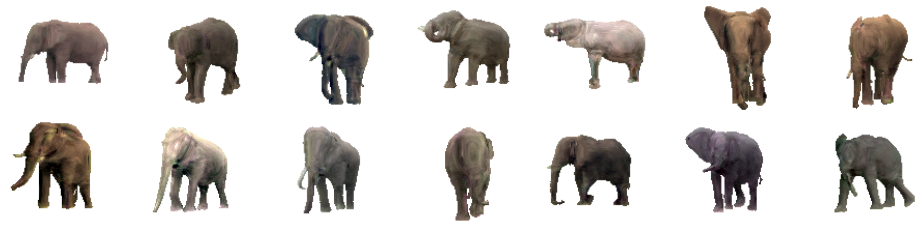
Figure 4.4: Reconstruction of the test set of (a) horses, (b) cats, (c) elephants and (d) facades with PPCA. The parameters of PPCA are shown in Table 4.2. CCCA reconstructions of the same set of images are shown in Figure 4.5.

several models to the horses dataset using different sets of hyperparameters:

$$F = 1, 4, 8, 12, 16, 20$$
$$K = 1, 8, 16, 24$$
$$M = 1500, 2000, 2500, 3000, 3500, 4000.$$

(a) Reconstructions of Horses with CCCA

(b) Reconstructions of Cats with CCCA



(c) Reconstructions of Elephants with CCCA

(d) Reconstructions of Facades with CCCA

Figure 4.5: Reconstruction of the test set of (a) horses, (b) cats, (c) elephants and (d) facades with CCCA. The parameters of CCCA are shown in Table 4.2.

Figure 4.6 shows the average and Figure 4.7 shows the variance of the mean per-pixel error of the reconstructions of the training and test sets of the horses dataset computed by the models.

As it can be seen in Figure 4.6, increasing the number of components reduces the error for both the training and test sets. As expected, when $K = 1$, *i.e.* the function $\phi[\cdot, \cdot]$ assigns a single value to each cluster of context vectors,

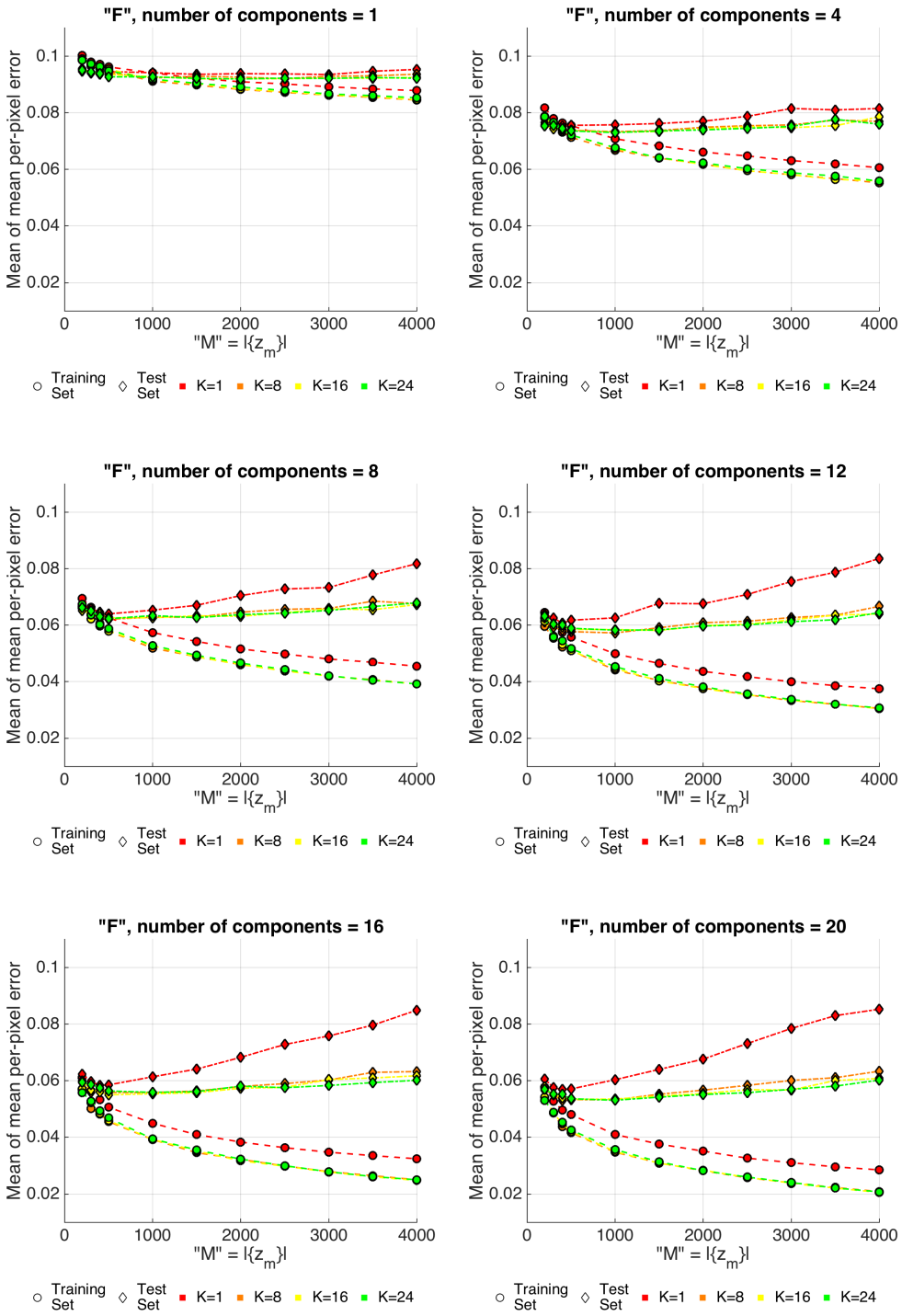


Figure 4.6: The average per-pixel error of the reconstruction of the training and test sets of the horses dataset computed by models with the different hyperparameters. Each subplot shows performance of 24 models with the same number of components and varying values of K and M.

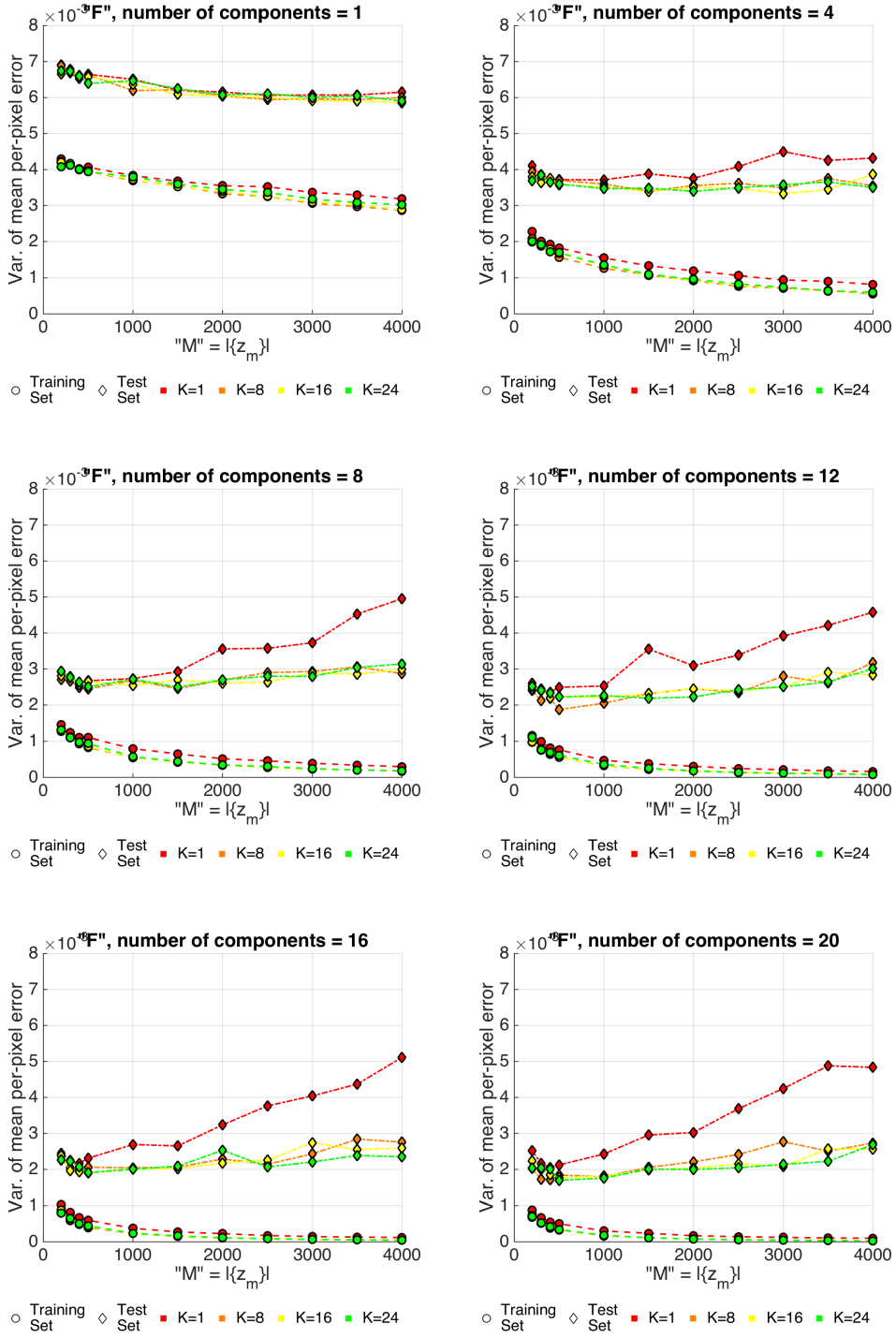


Figure 4.7: The variance of the per-pixel error of the reconstruction of the training and test sets of the horses dataset computed by models with the different hyperparameters. Each subplot shows the variance of the mean per-pixel error of the reconstructions of 24 models with the same number of components and varying values of K and M.

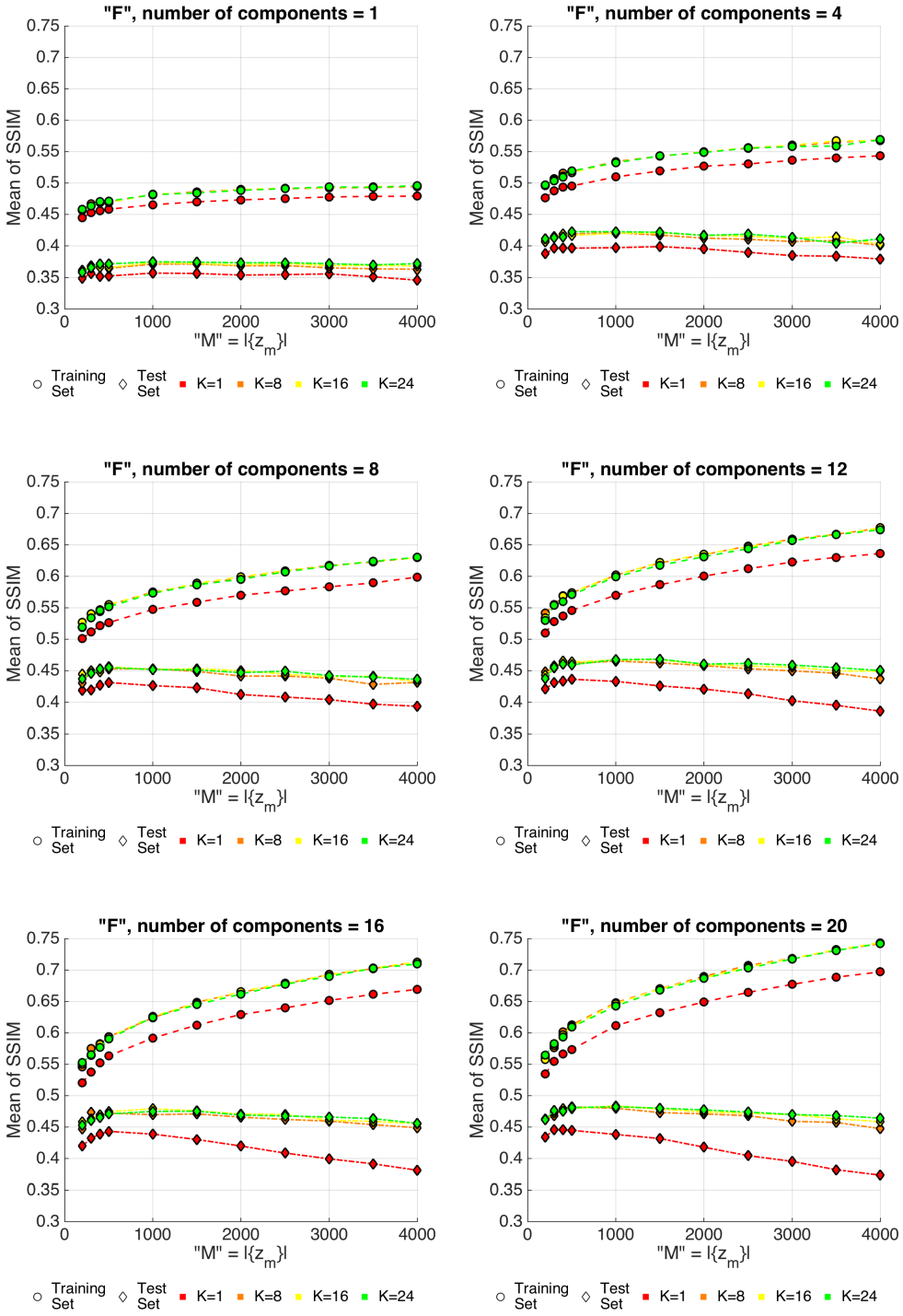


Figure 4.8: Structured similarity index measure (SSIM) [11] of the reconstruction of the training and test sets of the horses dataset computed by models with the different hyperparameters. Each subplot shows performance of 24 models with the same number of components and varying values of K and M. The SSIM' index is a value between -1 and 1, where value 1 is achieved when reconstructions are perfect.

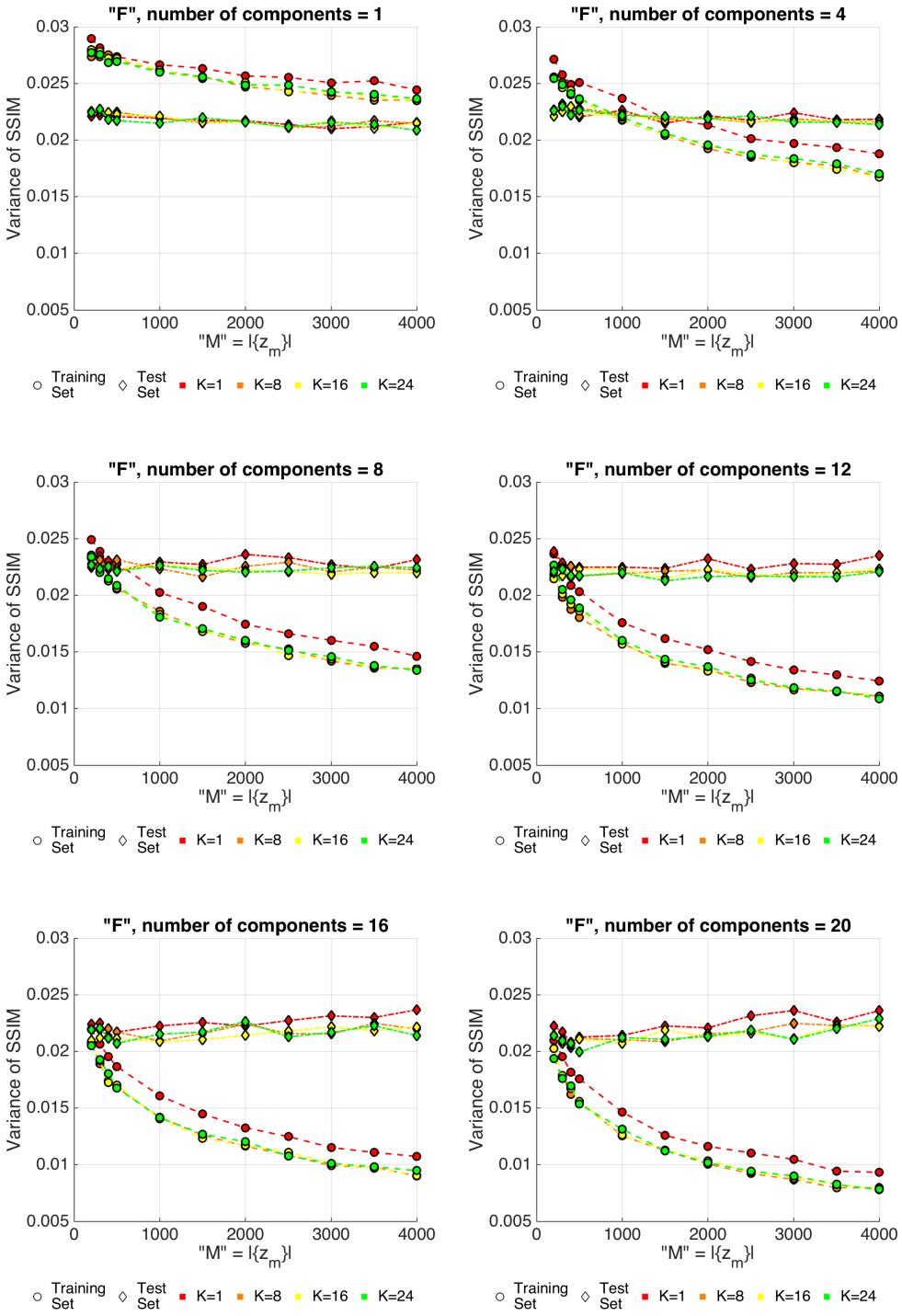


Figure 4.9: The variance of the SSIM score of the the reconstruction of the training and test sets of the horses dataset computed by models with the different hyperparameters. Each subplot shows the variance of the SSIM score of 24 models with the same number of components and varying values of K and M.

	C-CCA		PPCA	
Dataset	Training Set Mean Error	Test Set Mean Error	Training Set Mean Error	Test Set Mean Error
Horses	0.0258±0.0079	0.0547±0.0432	0.0268±0.0092	0.0886±0.0661
Cats	0.0280±0.0119	0.0511±0.0351	0.0315±0.0131	0.0548±0.0371
Elephants	0.0120±0.0031	0.0346±0.0163	0.0169±0.0048	0.0470±0.0220
Facades	0.0333±0.0094	0.0593±0.0264	0.0350±0.0198	0.0889±0.0330

Table 4.3: Mean Per-pixel Reconstruction Error (in RGB colorspace) for C-CCA and PPCA reported as `mean` ± `standard deviation`.

the model generalizes the least (the error on the test set is high), while larger values of K perform comparably. Interestingly, increasing M slowly increases the margin between reconstruction error on the training and test sets. Finally, increasing the number of components F reduces the error, which is similar to PCA.

Similarly, we evaluate the perceived image quality of reconstructions by comparing the original images with the reconstructions using Structural Similarity Index Measure [11]. Figure 4.8 shows mean SSIM with respect to the different hyperparameters of the model and Figure 4.9 shows the variance of SSIM of the different models. Notice, that SSIM is 1 when the images are identical, and -1 when they are completely different.

4.6.3 Appearance Transfer

The estimated function weights $\mathbf{h}_i$ and the colorspace rotation $\mathbf{R}_i$ and translation vector $\mathbf{t}_i$ of the image i of the dataset define the appearance of the visual object. So, for appearance transfer, we select another image j from the dataset, and reconstruct it by weighing the basis matrix $\mathbf{\Phi}_j$ with $\mathbf{h}_i$, and apply the colorspace transformation using $\mathbf{R}_i$ and $\mathbf{t}_i$. Figures 4.10 and 4.11 show examples.

Notice that computing a dense correspondence between some of these images is challenging. One of techniques for solving image alignment is SIFT flow [67]; Figure 4.12 illustrates how SIFT flow fails to compute a correspondence, as the pairs of images have animals in very different poses with multiple occlusions. On the other hand, C-CCA successfully transfers appearance from and to instances of horses that have different numbers of legs visible. The transforms are also consistent in terms of left-right axis, for example, the rightmost cat in Figure 4.11 has right leg black and left leg white, which is consistently transfered to different poses.

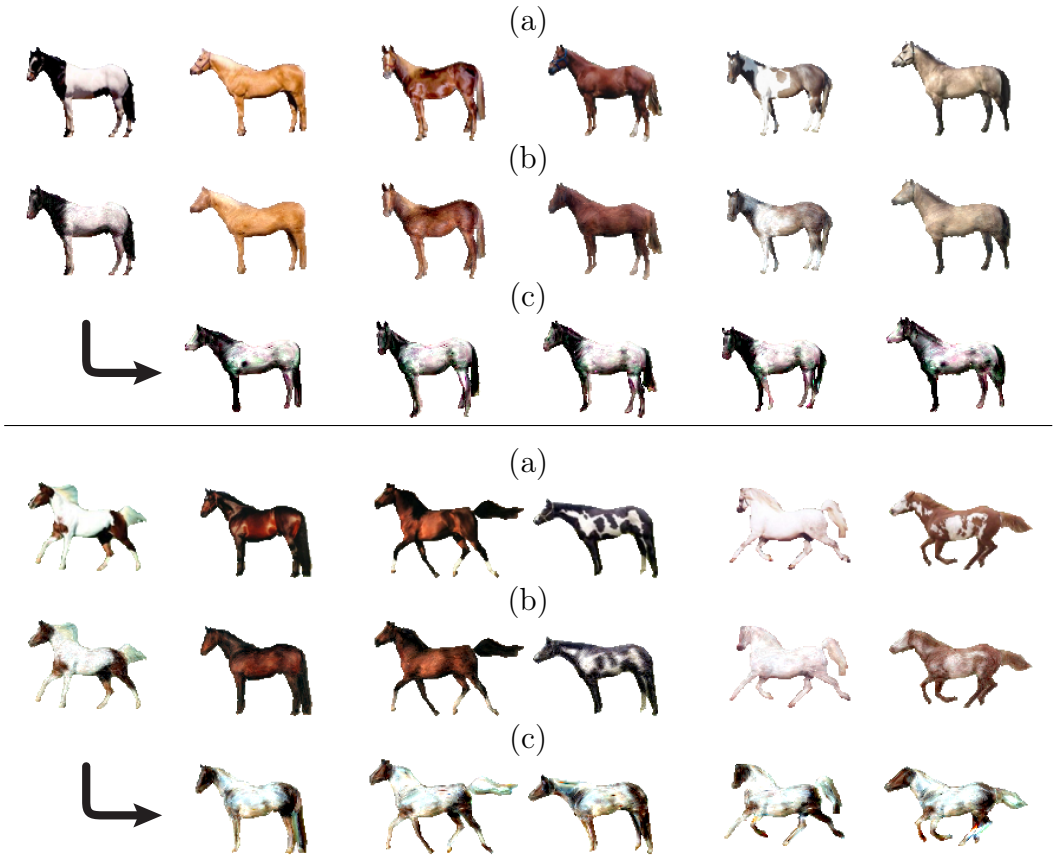


Figure 4.10: Appearance Transfer Results (Horses). (a) A subset of images of the training set. (b) Reconstruction of the images using the fitted subspace model with the estimated weights h_g. (c) Reconstruction of the images using the fitted subspace model with the fixed function weights h_g, rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ of the first (leftmost) image.

4.6.4 Structured Inpainting

We use the *test set* to do image inpainting. Half of the image's intensities are used to estimate regression function weights h_g and colorspace rotation matrix $\mathbf{R}_i$ and translation vector $\mathbf{t}_i$. We use the context vectors of *all* the pixels to reconstruct the whole image. Figures 4.13, 4.14 and 4.15 show the results.

Notice the difference between the reconstructed and the inpainted images. For example, in Figure 4.13, the horses in the first and fourth row of the left column have interesting change of the color from left to right. This color change is captured in the reconstruction of the full model. However, for inpainting, the observed left half of the image shows only brown color. Since there is no visible color variation, the model assumes that the whole horse is brown, as shown in the inpainted result. The same effect can be seen in the images of the fifth and seventh row of the left column in Figure 4.14.

Figure 4.11: Appearance Transfer Results (Cats): For each column, rows top to bottom: (1) An image of the training set. (2-4) Reconstruction of the image of *another* cat in a different pose by fixed function weights h_g, rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ of the top row image.

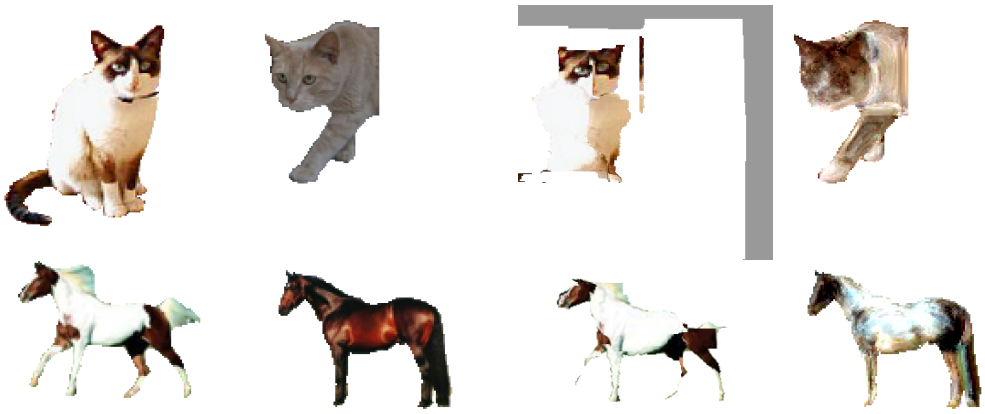


Figure 4.12: Left to Right: Source image. Target image. The warping of the source image onto the target image using the correspondence computed by SIFT flow [67]. The appearance transfer using C-CCA.

The inpainting results in Figure 4.16 of the Facades dataset demonstrate the importance of context vectors in our model. Due to the provided labeling, all windows in each of the facades have same context vectors. This results in the function $\phi[\cdot, \cdot]$ returning equivalent values for all pixels that were labeled as window. Hence, all windows in our reconstruction and inpainted images look the same.

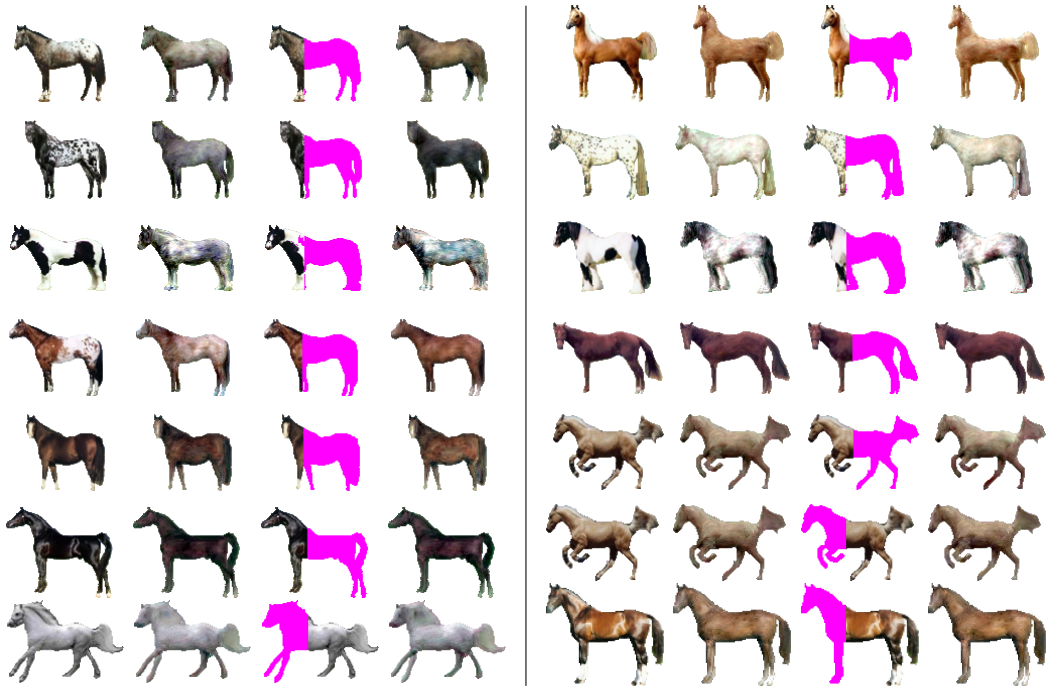


Figure 4.13: Image Inpainting Results for Horses (Test Set). In each column left to right: Image from the test set. Reconstruction of the image using the fitted model. Input for inpainting (context vectors of all pixels are known). Inpainted result.

4.7 Relation to Other Models

In this section we show how C-CCA is a generalization of the two models that are arguably most widely used in the Computer Vision field: Probabilistic PCA and Active Appearance Model.

4.7.1 Relation to Probabilistic PCA

Probabilistic Principal Component Analysis (PPCA) is a special case of the proposed model; for the case of a set of images where the context vectors match at each pixel so that $\mathbf{c}_{1j} = \mathbf{c}_{2j} \ldots = \mathbf{c}_{Ij}$, the mean vectors $\boldsymbol{\mu}_i$ and the basis matrices $\boldsymbol{\Phi}_i$ will be the same for every image i, and equation 4.2 becomes the generative model for PPCA:

$$\mathbf{x}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \boldsymbol{\epsilon}_i \ . \tag{4.21}$$

It follows that the proposed model is a generalisation of PPCA where the mean vector $\boldsymbol{\mu}$ and the principal components $\boldsymbol{\Phi}$ depend on the context vectors for the given image $\{\mathbf{c}_{ij}\}_{j=1}^{J}$. Hence, C-CCA creates a different but related PPCA model for each image in the set.

Figure 4.14: Image Inpainting Results for Cats (Test Set). In each column left to right: Image from the test set. Reconstruction of the image using the fitted model. Input for inpainting (context vectors of all pixels are known). Inpainted result.

4.7.2 Relation to Active Appearance Model

In this subsection we show that Active Appearance Model [8] is another special case of C-CCA when a specific form of $\phi[\cdot,\cdot]$ is adopted. We start by describing AAM.

AAM The AAM assumes that the input consists of a set of images $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{I,J}$ and a corresponding set of x-axis and y-axis coordinates of the fiducial points $\{\mathbf{u}_{iq}\}_{i=1,q=1}^{I,Q}$. (For example, for faces, the fiducial points are points such as corners of the eyes, the tip of the nose, chin, *etc.*)

Next, the mean coordinate of each of the fiducial points is computed: $\{\bar{\mathbf{u}}_q\}_{q=1}^{Q}$. It is assumed that these coordinates define the mean shape of the object, which serve as a "common template" for the images of the dataset.

The mean coordinates $\{\bar{\mathbf{u}}_q\}_{q=1}^{Q}$ are used as vertices of a mesh with triangular

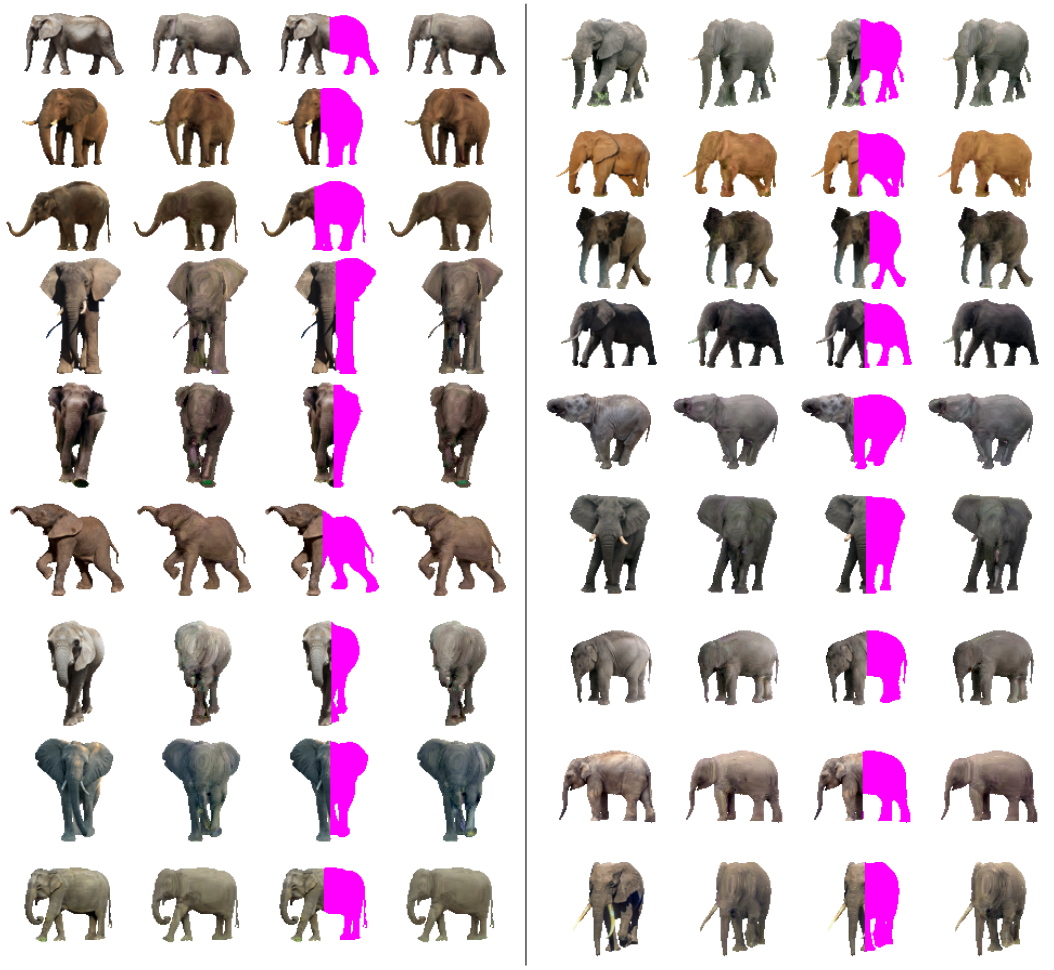


Figure 4.15: Image Inpainting Results for Elephants (Test Set). In each column left to right: Image from the test set. Reconstruction of the image using the fitted model. Input for inpainting (context vectors of all pixels are known). Inpainted result.

faces computed by Delaunay triangulation [241] or some other variation of it. Thus, each pixel of the common template has corresponding spatial coordinates, the triangle of the mesh and barycentric coordinates to the vertices of the corresponding triangle.

All images are warped to this common template. This is done with a piecewise affine warp that maps each triangle of the mesh from image i to the common template. So, for each image i and its triangle τ, AAM computes the affine transformation that maps coordinates of the vertices of the triangle τ to the coordinates of the corresponding vertices in the common template. Hence, the coordinates of the pixel and the fiducial points $\{\mathbf{u}_{iq}\}_{i=1,q=1}^{I,Q}$ are used to *deterministically* compute the pixel coordinates in the common template.

Once all images are warped to the same template, the vectorized represen-

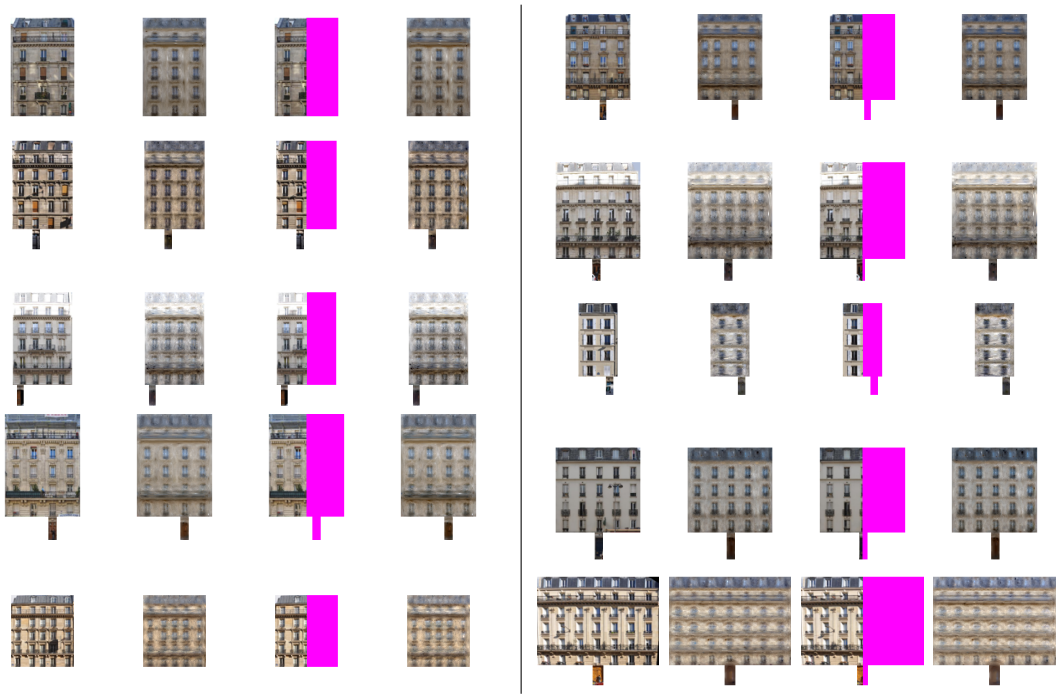


Figure 4.16: Image Inpainting Results for Facades (Test Set). In each column left to right: Image from the test set. Reconstruction of the image using the fitted model. Input for inpainting (context vectors of all pixels are known). Inpainted result.

tation of the warped images can be used as input to a subspace model such as PCA. The subspace models the appearance of the object. To fit the learned model to a new image and its set of fiducial points, one must compute the transformation from the common template to the new image.

C-CCA Next, we show that AAM is equivalent ot C-CCA when a specific $\phi[\cdot,\cdot]$ is adopted. As it was mentioned, we define the form of $\phi[\cdot,\cdot]$ as in equation 4.14.

So, image i has fiducial coordinates $\{\mathbf{u}_{iq}\}_{i=1,q=1}^{I,Q}$. Furthermore, the j-th pixel of image i has a pixel intensity of x_{ij} and it's x-axis and y-axis coordinates $\mathbf{v}_{ij}$. We define its context vector as

$$\begin{aligned}\mathbf{c}_{ij} = [&\mathbf{v}_{ij}^T \\ &\mathbf{u}_{i1}^T \dots \mathbf{u}_{iQ}^T \\ &\sqrt{|\mathbf{v}_{ij} - \mathbf{u}_{i1}|^2} \dots \sqrt{|\mathbf{v}_{ij} - \mathbf{u}_{iQ}|^2}]^T \ ,\end{aligned} \tag{4.22}$$

which is a concatenation of pixel's coordinates $\mathbf{v}_{ij}$ (2 values), the coordinates of the fiducial points ($2 \times Q$ values) and Euclidean distances to all of the fiducial

points (Q values).

Let $\boldsymbol{\theta}_f$ be a vector of size $J \times 1$, where J is the number of pixels in the "common template". First, assume that $\mathbf{a}\left[\mathbf{c}_{ij}\right]$ returns a vector of size $J \times 1$ that consists of zeros and a single 1.

It follows that by using the mean coordinates $\{\bar{\mathbf{u}}_q\}_{q=1}^{Q}$ and the Delaunay triangulation, one can define such $\mathbf{a}\left[\mathbf{c}_{ij}\right]$ that *deterministically* defines which index of the returned vector has value of 1. This effectively maps every pixel of the input image to the common template similarly to AAM.

Indeed, the first $2 + 2 \times Q$ values of $\mathbf{c}_{ij}$ (the pixel's coordinates and the coordinates of the fiducial points) can be used to determine the triangle of the mesh's triangulation that the pixel belongs to. Furthermore, the last Q values of $\mathbf{c}_{ij}$ (Euclidean distances to all of the fiducial points) can be used to compute the barycentric coordinates of the pixel. The barycentric coordinates can be used to deterministically define which of the pixels in the common template x_{ij} corresponds to, which in turn defines which index of the returned vector have value of 1.

Hence, such $\mathbf{a}\left[\mathbf{c}_{ij}\right]$ maps images to the common template similarly to AAM. Thus, $\boldsymbol{\theta}_f$ are equivalent to the components of PPCA.

By a similar line of reasoning one can show that Layered AAM [9] is also a special case of C-CCA.

4.7.3 Multifactor Models

In this subsection we analyze the differences between C-CCA and Multifactor Models.

Bilinear [242, 25] and multilinear [243, 244] methods model the data as a product of multiple factors, such as style, identity, motion phase, *etc*. The Multifactor Gaussian Process Model [245] is a non-linear generalization of the multilinear models which uses non-linear basis functions. For example, let us consider a case, where the data has two modes of variation: the appearance and the style, so the captured data is modeled as a function of the appearance represented with vector $\hat{\mathbf{h}}$ and the style $\hat{\mathbf{c}}$. Here, we choose the model to be linearly dependent on the appearance and non-linearly dependent on the style

$$x_{ij} = f_j[\hat{\mathbf{h}}_i; \hat{\mathbf{c}}_i] = \sum_{f=0}^{F} \hat{\phi}_{fj}[\hat{\mathbf{c}}_i]\hat{h}_{if} + \boldsymbol{\epsilon}_i \ , \tag{4.23}$$

where $\boldsymbol{\epsilon}_i$ is the Gaussian noise and $\hat{\phi}_{fj}[\hat{\mathbf{c}}_i]$ is a non-linear function of style. The

linear and non-linear dependencies of different factors in Multifactor Gaussian Process Models are chosen a priori with appropriate kernels and are motivated by the knowledge of the intrinsic factors of the data.

The Multifactor Gaussian Process Models are similar to C-CCA, which also models data with non-linear basis functions. However, there is an important distinction. The components in Multifactor Gaussian Process Models are dependent on the style factor $\hat{\mathbf{c}}_i$, which is *global* to all dimensions of each of the data points. Or,

$$\hat{\mathbf{c}}_{i1} = \hat{\mathbf{c}}_{i2} = \ldots = \hat{\mathbf{c}}_{iJ} \ . \tag{4.24}$$

On the other hand, in C-CCA the context vector of the pixel $\mathbf{c}_{ij}$ can encode both *local* and *global* information

$$x_{ij} = \sum_{f=0}^{F} \phi_f[\mathbf{c}_{ij}, \cdot] h_{if} + \boldsymbol{\epsilon}_i \ . \tag{4.25}$$

This distinction has a major consequence: C-CCA can use local context information for implicit alignment, which allows it to model data that is not aligned, or even structured.

4.7.4 Alignment with Components

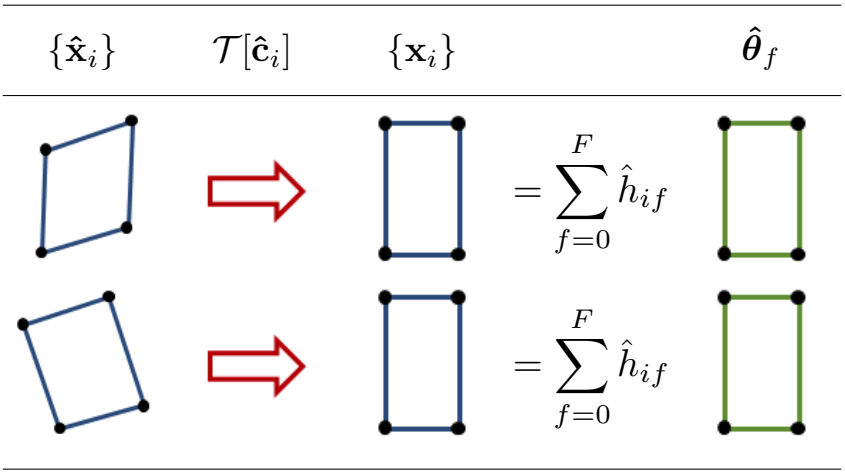
In C-CCA, the form of the components depends on the form $\phi[\cdot, \cdot]$. If we choose the representation discussed in section 4.4.5, one can show that

$$\boldsymbol{\theta}_f = \sum_{i=1}^{I} \lambda_{fi} \mathcal{T}[\mathbf{c}_i] \mathbf{x}_i \ , \tag{4.26}$$

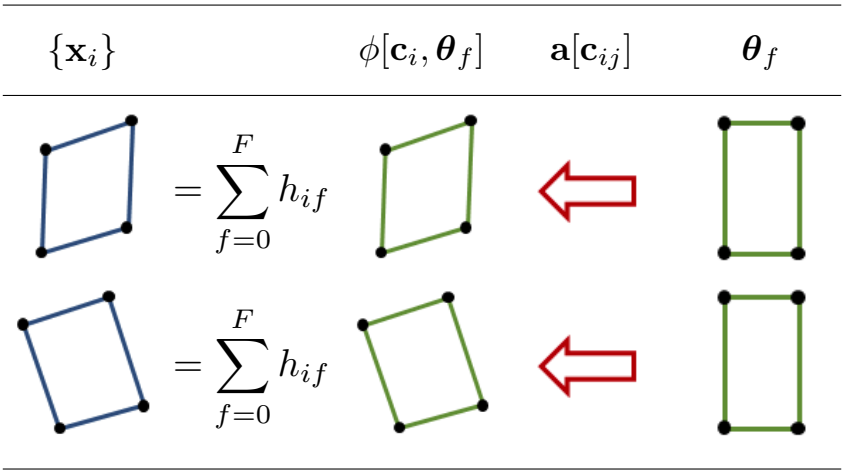
where $\mathcal{T}[\cdot]$ transforms the training data point $\mathbf{x}_i$ using the corresponding context vector $\mathbf{c}_i$.

This perspective provides an interesting interpretation of the model, which we discuss in this subsection.

In the traditional pipeline of fitting an appearance model to the image set, the first stage is the preprocessing of the data with the registration or alignment methods, and the second stage is the fitting of parameters of the parametric model. For example, AAMs use the keypoint annotations to warp the images to a common shape template. In general, the original data $\hat{\mathbf{x}}_i$ is transformed with some deformation $\mathcal{T}[\cdot]$, which is chosen for each image $\hat{\mathbf{x}}_i$ using some annotation data $\hat{\mathbf{c}}_i$. So, *the images are transformed to align* with each other so that the data can be fitted with a parametric model. For example,



(a) Illustration of Alignment in AAMs.



(b) Illustration of Alignment in C-CCA.

Figure 4.17: Illustration of the difference between AAMs and C-CCA from the perspective of alignment in image space and component space.

in AAMs the transformed data is modeled with PCA as

$$\mathcal{T}[\hat{\mathbf{c}}_i]\hat{\mathbf{x}}_i = \sum_{f=0}^{F} \hat{\boldsymbol{\theta}}_f \hat{h}_{if} \ , \tag{4.27}$$

where $\hat{h}_{if}$ are the component weights and $\hat{\boldsymbol{\theta}}_f$ are the principal components. See Figure 4.17(a) for an illustration of equation 4.27.

In contrast, C-CCA aligns and registers the data with the functions $\phi[\cdot, \cdot]$

$$\mathbf{x}_i = \sum_{f=0}^{F} \phi[\mathbf{c}_{ij}, \boldsymbol{\theta}_f] h_{if} = \sum_{f=0}^{F} \mathbf{a}[\mathbf{c}_{ij}] \boldsymbol{\theta}_f h_{if} \ , \tag{4.28}$$

which can be interpreted as *transforming the components to align with the input image* using transformation $\mathbf{a}[\mathbf{c}_i j]$. See Figure 4.17(b) for an illustration of equation 4.28. Hence, there is no common shape template in C-CCA, because the representation is transformed to align with each of the input images. The difference between AAMs and C-CCA from the discussed perspective is illustrated in Figure 4.17.

This alignment in components assumes that pixels from different images which have similar context vectors should also have similar values for each component function $\phi[\cdot, \cdot]$. The appearance variation is modeled by the weights of the components $\mathbf{h}_i$.

4.8 Conclusion

We have proposed a novel generative subspace model that exploits additional information, such as segmentation or semantic labeling, to model the appearance of visual objects. The "component functions" learned by our model are conditioned on the per-pixel context vectors that were computed using additional information. This enables sharing of appearance information in a complex, non-linear way. We derive an efficient iterative learning algorithm by restricting the class of component functions. We demonstrated that C-CCA can successfully model challenging datasets that have complex occlusions highly-deformable visual objects and varying numbers of parts. We reported both qualitative and quantitative results of our model with examples of structured inpainting and appearance transfer. Unsurprisingly, C-CCA outperforms PPCA in the structured inpainting task at a fraction of learned parameters. We discuss the relation of C-CCA to other popular models, and show that C-CCA is a generalization of PPCA and Active Appearance Models with an implicit alignment mechanism. Finally, C-CCA demonstrates that there are viable methods of appearance modeling besides the traditional pipeline with explicit alignment step.

4.9 Discussion

In this work we naively normalize the context vectors for the k-NN regression learner; however, a better approach would be to learn the weights for the dimensions of the context vector using the training data or apply distance learning techniques.

Furthermore, our model is agnostic to the nature of the context vectors, so it may be possible to learn the context from the data directly, without the

manual labeling; *i.e.* learn a set of filters that maximize the reconstruction quality or use the probability distribution of labels learned by a part classifier. Unsupervised methods that learn a set of discriminative patches could be directly used to learn a set of good filters.

Finally, one could explore other classes of functions that could be used for $\phi[\cdot,\cdot]$. For example, the full model (equations 4.5 and 4.5) could be optimized with a gradient descent. Hence, the function $\phi[\cdot,\cdot]$ can be modeled with a Neural Network.

Chapter 5

Synthesizing Images

"And between the art of creating and the art of repeating there is a great difference. To create means to live, forever creating newer and newer things."

—Kazimir Malevich

"Между искусством творить и искусством повторить — большая разница. Творить — значит жить, вечно создавая новое и новое."

Казимир Малевич

In this chapter we investigate automatic synthesis of a novel image given a set of semantic part masks. Our pipeline has two stages: (i) drawing a sample from a globally-coherent parametric model conditioned on the semantic part masks; (ii) post-processing of the drawn sample with a locally-coherent non-parametric model.

In the first stage, we use Context-Conditioned Component Analysis described in Chapter 4 as the parametric model that is learned from a library of images with respective object masks and semantic part labels. We show how to sample the component weights $\mathbf{h}_i$ of C-CCA and the colorspace parameters $\mathbf{R}_i$ and $\mathbf{t}_i$.

Since the learned model is low-dimensional, the sampled images lack high-frequency details. We address this problem by the second stage: hallucinating high-frequency details by stitching together patches extracted from the training image set (also used for training C-CCA) that match the target C-CCA sample. The patch correspondence problem is formulated as a Conditional Random Field minimization, where the unary term incorporates the appearance of the patch and its semantic part label. To solve the optimization problem efficiently, we extend the PatchMatch Belief Propagation algorithm [12] to

search over a collection of images by precomputing a graph of inter-image correspondences [230].

We demonstrate synthesis results of our method on datasets of horses and elephants.

5.1 Introduction

Synthesizing images is the main research problem of the Computer Graphics field. The rendering approach relies on simulating interactions of physical elements of the scene, such as reflectance properties of the materials, geometry of surfaces, lighting, camera parameters, *etc*. These properties must be specified by trained artists, which is a tedious and time-consuming process.

Besides rendering techniques, there are methods that try to learn the appearance of the visual object by analyzing a library of example images. In this chapter we address the problem of synthesizing images of highly-deformable, non-structured visual objects.

As discussed previously, there are two, principally different, data-driven approaches to the problem of image synthesis: generative parametric models and non-parametric models.

The generative parametric models build a parametric low-dimensional representation of a visual object from a library of images. The generative models can infer the appearance parameters from an image, as well as reconstruct an image from the appearance parameters. This allows one to generate novel images by drawing samples of the parametric model with random appearance parameters. The intrinsic property of parametric models is the dimensionality reduction, which results in the sampled images to lack high-frequency details as shown in Figure 5.4.

The non-parametric models copy patches from the library of images, such that the overlaps of the copied patches have local coherence. This method has been successfully used for texture synthesis [150]. Blending adjacent patches in the gradient domain [203] was shown to further improve the realism of the result. Later, Darabi *et al*. [159] extended the patch-based synthesis method to combine inconsistent textures in a realistic way. However, these methods lack the notion of global coherence. Hence, inputs of tens or hundreds of images may produce combinations of patches that are a locally-coherent, but globally-incoherent. Examples of such combination are "Frankenstein" images, see Figure 2.18.

We further discuss "Visio-lization" [1] method that combined parametric

and non-parametric models, however, we refer the reader to Chapter 2 for a thorough review of other related work.

Mohammed *et al.* [1] combined parametric and non-parametric models for the problem of generating images of faces. The images are generated in two stages. First, the parametric model is sampled for a new image. Then, the sampled image is divided into a grid and the non-parametric model copies patches of the library to each cell of the grid. The patches are chosen such that they have local coherence with each other and they are close to the color values of the sampled image. This pipeline complements the two models. Indeed, constraining the non-parametric model to be close to the samples of the parametric model adds global coherence to the results of the non-parametric model. Furthermore, applying non-parametric model hallucinates high-frequency details on the blurry sample of the parametric model. Mohammed *et al.* [1] use a Probabilistic Principal Component Analysis as the global parametric model to samples images of faces. PPCA works well for visual objects such as faces; as they have fixed structure, they are already aligned to each other, and they do not have significant deformations. However, it cannot be readily applied to non-structured highly-deformable visual objects, such as horses and elephants, as shown in Figure 1.4, 1.5 or 4.4 (a) and (c). Since the faces are prealigned, the patch correspondence problem of the non-parametric model is straightforward: for any cell of the grid, suitable patches are searched in the corresponding cell location, without significant rotations. However, this assumption does not hold for deformable objects. For example, consider the patch corresponding to the head of a horse, it can be located close to the bottom of an image, if the horse is eating grass, or, it can be located close to the top of an image, if the horse is galloping.

Similarly to "Visio-lization", we adopt the approach of combining parametric and non-parametric models by first generating a globally-coherent image T by sampling the parametric model and then improving the visual fidelity of T with a non-parametric model.

We utilize Context-Conditioned Component Analysis as the parametric model. In Chapter 4 we formulated C-CCA as a generative parametric model of the appearance of the visual object conditioned on the shape information. We demonstrated that it can successfully model the appearance of the visual object in such tasks as structured inpainting and appearance transfer. In this chapter we randomly sample the learned C-CCA model to generate images of the visual object that have novel appearance using semantic part labels and

segmentation masks of the test set images. To this end, we learn the probability distribution of the appearance parameters to generate realistic samples. We describe the sampling in section 5.2.

In the second stage of our pipeline, the non-parametric model hallucinates high-frequency details of the C-CCA sample. This is done by stitching high-quality patches extracted from the training set images (also used for training C-CCA) that match the target C-CCA sample. The patch correspondence problem is formulated as a minimization of a Conditional Random Field. Fast search of patches is a challenging problem when a dataset consists of hundreds of images, each of which contains thousands or millions (taking into account multiple scales and rotations) of candidate patches. Hence, we extend the state-of-the-art patch correspondence method, PatchMatch Belief Propagation [12], to find suitable patches over a collection of images. To perform the patch search efficiently, we precompute a graph of inter-image correspondences over all pairs of images in the dataset. This non-parametric model is described in section 5.3.

In section 5.4 we show results of the full model on two challenging datasets.

5.2 Global Parametric Model

In this section we describe how we sample the learned Context-Conditioned Component Analysis model for novel appearances. The learning is described in Chapter 4.

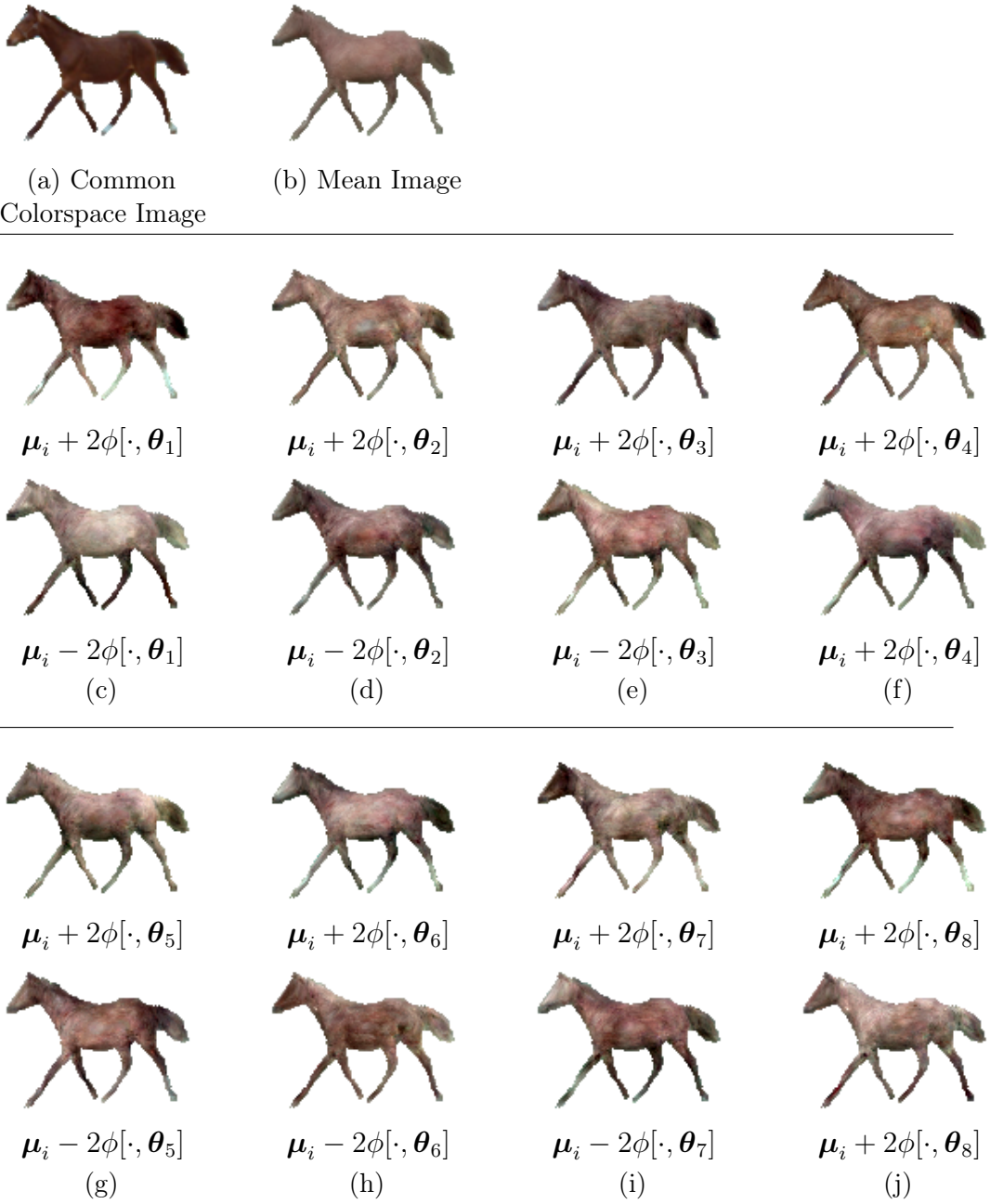
5.2.1 Overview

We fit Context-Conditioned Component Analysis to the images of the dataset as described in sections 4.4 and 4.5. So, the appearance of the object in each image is described by $F \times 1$ vector $\mathbf{h}_i$, which represents component weights, and 3×3 colorspace rotation matrix $\mathbf{R}_i$ and 3×1 colorspace translation vector $\mathbf{t}_i$. The component weights have an independent standard Gaussian prior

$$Pr(\mathbf{h}_i) = \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \; . \tag{5.1}$$

Figures 5.1 and 5.2 visualizes the appearances in the common colorspace learned by C-CCA with $F = 8$ components. Notice, how the appearance changes for a component are consistent between the two horses that are in very different poses. For example, the component $\phi[\cdot, \boldsymbol{\theta}_1]$ controls the brightness of the head relative to the body of the horse in both examples.

Unfortunately, the colorspace parameters $\mathbf{R}_\star$ and $\mathbf{t}_\star$ do not have a prior



(a) Common Colorspace Image (b) Mean Image

$\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_1]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_2]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_3]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_4]$

$\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_1]$ $\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_2]$ $\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_3]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_4]$

(c) (d) (e) (f)

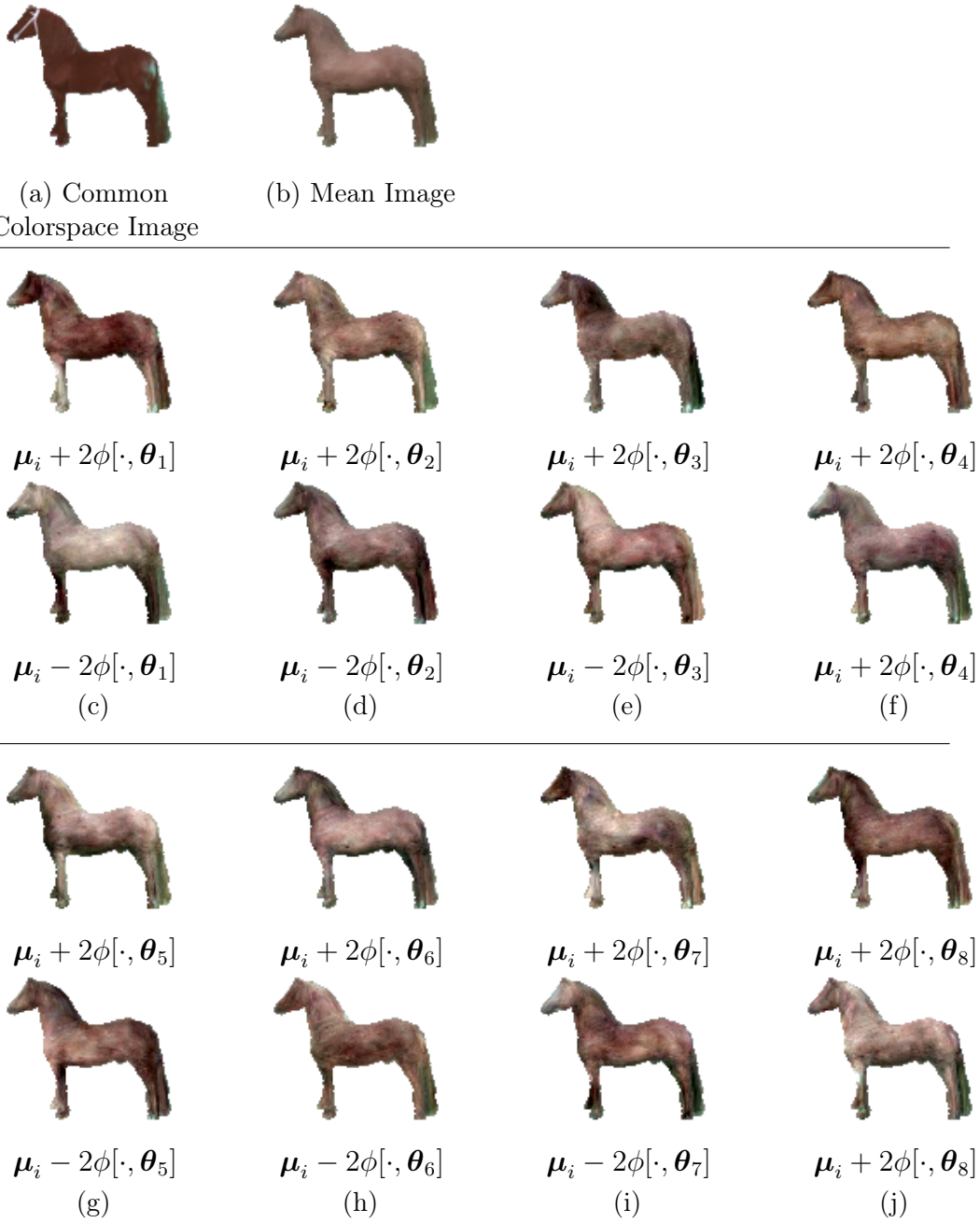
$\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_5]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_6]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_7]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_8]$

$\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_5]$ $\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_6]$ $\boldsymbol{\mu}_i - 2\phi[\cdot, \boldsymbol{\theta}_7]$ $\boldsymbol{\mu}_i + 2\phi[\cdot, \boldsymbol{\theta}_8]$

(g) (h) (i) (j)

Figure 5.1: Illustration of the appearance manifold learned by C-CCA with $F = 8$ components. (a) A testset image with the colorspace rotated and translated to the common colorspace. (b) Mean image. (c)-(j) Moving along the component from the mean by two standard deviations in positive and negative directions.

that we can use to sample. So, we model colorspace parameters with a Gaussian Process Latent Variable Model [36]. The colorspace parameters for each image are 12×1 vectors that are concatenations of vectorized forms of the colorspace



(a) Common Colorspace Image (b) Mean Image

(c) (d) (e) (f)

(g) (h) (i) (j)

Figure 5.2: Illustration of the appearance manifold learned by C-CCA with $F = 8$ components for a horse in a different pose (see Figure 5.1). (a) A testset image with the colorspace rotated and translated to the common colorspace. (b) Mean image. (c)-(j) Moving along the component from the mean by two standard deviations in positive and negative directions.

rotation matrix $\mathbf{R}_i$ and colorspace translation vector $\mathbf{t}_i$. We fit a 2-dimensional latent representation of the colorspace parameters. We acknowledge that linear interpolation of rotation matrices may not produce an orthogonal rotation

matrix, so, a different representation of the rotation matrix, such as quarternion representation in Shoemake [246], is left as future work.

5.2.2 Sampling

We assume that we have a semantic part label map, which is used to compute context map $\mathbf{c}_\star$, either from a test set data or from some generative shape model. While samples of the generative model of contours introduced in Chapter 3 (section 3.4.2) are sufficiently detailed to serve as a guide to the user, they, unfortunately, lack high frequency details and cannot be readily applied in this method. Hence, in this chapter we use context maps from the test set of the data. The context map is used to evaluate component functions $\phi[\cdot,\cdot]$. Next, we draw random samples of the component weights $\mathbf{h}_\star$ from a standard Gaussian distribution. The component functions $\phi[\mathbf{c}_\star,\cdot]$ are weighted by the random component weights $\mathbf{h}_\star$ and linearly combined to generate a novel image T in the "common" colorspace. Examples of the generated samples are shown in Figure 5.3.

We also sample the GPLVM latent space from a uniform distribution, such that the variance of the drawn sample is within a threshold. The random sample can be decomposed into colorspace parameters $\mathbf{R}_\star$ and $\mathbf{t}_\star$. We normalize the sample such that $\mathbf{R}_\star$ is an orthogonal rotation matrix. These colorspace parameteres are applied to the output of the local model to generate the final result. For illustration, we show the changes in color by applying the color transformation to samples of C-CCA in Figure 5.4. Notice the difference between the original image of the test set and the random appearance generated by the Global Model using the context of the original image.

5.3 Local Non-parametric Model

In this section we describe how we hallucinate high-frequency details on top of the image T. This method has two important procedures: (i) finding suitable patches from multiple images of the training set and (ii) blending the patches together.

The first procedure, the search of the patches, is known as patch correspondence problem in computer vision and graphics. To achieve high visual fidelity, we find patches that minimize an objective function that has global constraint terms associated with image T and local constraint terms that ensure local visual coherence. In subsection 5.3.1 we discuss the patch correspondence problem and the state-of-the-art approaches in the field.

Figure 5.3: Samples of C-CCA in the common colorspace. Left: Image from the test set. Right: 5 random samples of C-CCA in Common Colorspace, *i.e.* $\mathbf{h}_\star$ is random, $\mathbf{R}_\star$ is identity matrix and $\mathbf{t}_\star$ is zero vector. C-CCA components are conditioned on the context of the image on the left.

The key idea of detail hallucination is that patches of the training images that are close to the patches of the blurry image T contain high-frequency details. The way of combining found patches also affects the generated high-frequency details. We discuss alternatives of blending in subsection 5.3.2.

Naturally, we transform images of the training set to the common colorspace and work with patches in this colorspace. Hence, after the high-frequency details are hallucinated, we transform the colors with random samples of $\mathbf{R}_\star$ and $\mathbf{t}_\star$. Figure 5.5 visualizes the pipeline.

Figure 5.4: Samples of $\mathbf{R}_\star$ and $\mathbf{t}_\star$ colorspace parameters applied to the samples of C-CCA. Left: Image from the test set. Right: 5 random samples of $\mathbf{R}_\star$ and $\mathbf{t}_\star$ colorspace parameters drawn from GP-LVM. In each row, different $\mathbf{R}_\star$ and $\mathbf{t}_\star$ colorspace parameters are applied to one random draw of $\mathbf{h}_\star$ of C-CCA. C-CCA components are conditioned on the context of the image on the left.

5.3.1 Patch Correspondence Problem

The patch correspondence problem is identifying for each patch of the target image a corresponding patch from one or more source images according to a cost function. Unfortunately, finding suitable patches from a collection of images is a challenging problem. Indeed, if the patches can have rotation and scale variations, then the number of patches that can be extracted from a few hundred images is in the millions, even when the source images have low resolution. An exhaustive patch-to-patch comparison is infeasible, so, randomized and

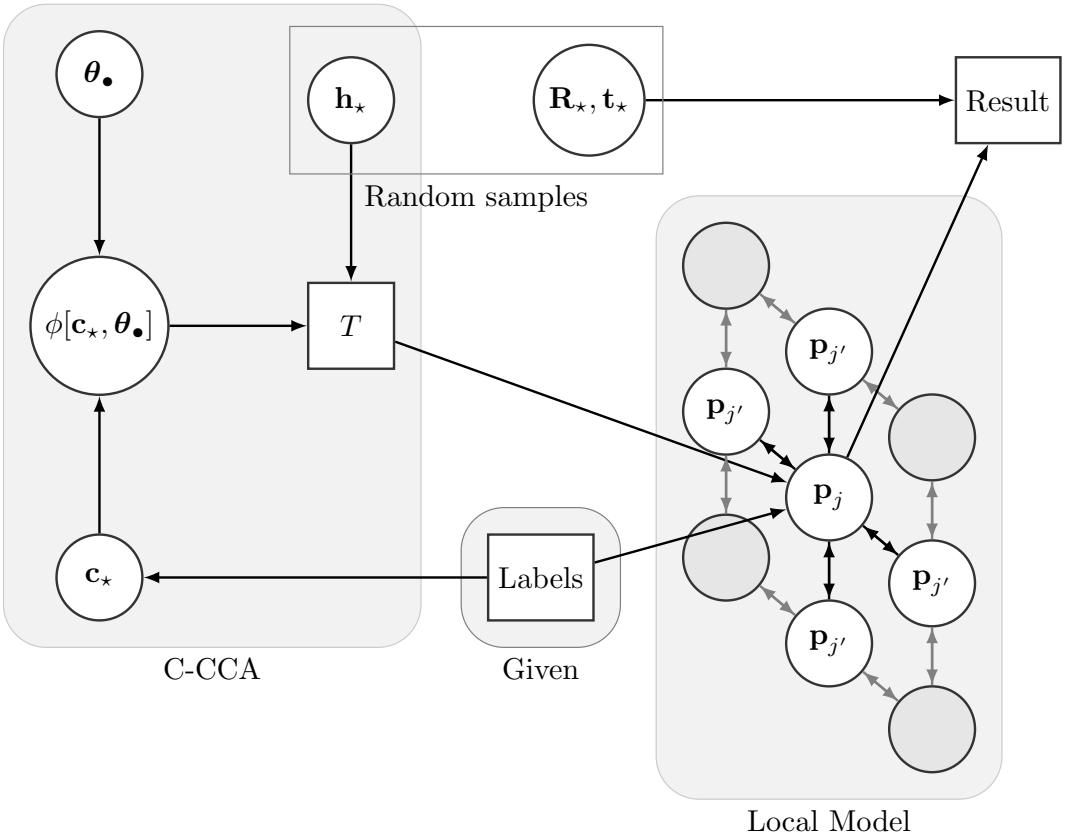


Figure 5.5: Visualization of the image synthesis pipeline. Circles represent variables, and rectangles represent images. We assume that the label map is given, either from the test set or a shape model. The label map produces context vectors $\mathbf{c}_\star$ for C-CCA. The C-CCA generates image T by drawing a random sample of $\mathbf{h}_\star$. The local model is defined over a grid of pixels and finds patches that are globally coherent (*i.e.* the patches corresponding to $\mathbf{p}_j$ agree with image T and the label map) and locally coherent (*i.e.* patch correspondences $\mathbf{p}_j$ agree with 4-neighbors $\mathbf{p}_{j'}$). The result is generated by stitched patches together and transforming the colorspace with random samples of $\mathbf{R}_\star$ and $\mathbf{t}_\star$.

approximate algorithms are used [12, 154, 158, 230]. So, we start by explaining the state-of-the-art PatchMatch and PatchMatch Belief Propagation algorithms. Then we show how we can speed up the search over multiple source images by precomputing the graph of inter-image correspondences.

First, we define some of the terms in the patch correspondence problem. We parameterize a patch coordinate (*i.e.* pointer) by a vector

$$\mathbf{p} = [x, y, \alpha, s, i] \,, \tag{5.2}$$

where x and y are spatial coordinates of the center of the patch, α is the

rotation of the patch around its center, s is the scale of the patch (*i.e.* scale factor of the coordinate grid of the patch) and $i = 1 \dots I$ is the index of an image from the source image set.

Let the vectorized form of the values of the pixels extracted from an $n_\mathrm{p} \times n_\mathrm{p}$ patch with parameters $\mathbf{p}$ be represented by a function

$$\boldsymbol{\rho}_\mathrm{color}[\mathbf{p}] = \boldsymbol{\rho}_\mathrm{color}([x, y, \alpha, s, i]) \ \in \mathcal{R}^{3{n_p}^2}. \tag{5.3}$$

The patches have a corresponding mask of size $n_\mathrm{p} \times n_\mathrm{p}$, which is 0 for pixels that are not defined, and 1 otherwise. The undefined pixels are pixels that correspond to the background of images.

$$\boldsymbol{\rho}_\mathrm{mask}[\mathbf{p}] = \boldsymbol{\rho}_\mathrm{mask}([x, y, \alpha, s, i]) \in \{0, 1\}^{{n_p}^2} \ . \tag{5.4}$$

The mask is important for patches on the boundaries of objects, so that the comparison of patches can ignore undefined pixels.

In a similar fashion, we can define the vector representation of the semantic part labels of the pixel centered at $\mathbf{p}$ as

$$\boldsymbol{\rho}_\mathrm{label}[\mathbf{p}] = \rho_\mathrm{label}\left[[x, y, i]\right] \ \in \mathcal{R}^{\texttt{number of parts}}, \tag{5.5}$$

where $\rho_{label}[\mathbf{p}]$ returns a vector of size $\texttt{number of parts} \times 1$ with values 1 and 0 at corresponding dimensions associated with the semantic part labels.

PatchMatch The objective of the PatchMatch algorithm is to find a matching patch $\boldsymbol{\rho}_\mathrm{color}[\mathbf{p}]$ from the i-th source image for every patch of the target image τ. Formally, the objective is to minimize some unary cost. For example, let the unary cost be

$$\begin{aligned}\psi_\mathrm{color}\left[\tau, \mathbf{p}_j\right] = ||\boldsymbol{\rho}_\mathrm{mask}\left([x_{\tau,j}, y_{\tau,j}, 0, 1, \tau]\right) \odot \\ \left(\boldsymbol{\rho}_\mathrm{color}\left([x_{\tau,j}, y_{\tau,j}, 0, 1, \tau]\right) - \boldsymbol{\rho}_\mathrm{color}\left[\mathbf{p}_j\right]\right)||^2 \ ,\end{aligned} \tag{5.6}$$

over all axis-aligned, standard scale patches of the target image τ, so $\forall j = 1 \dots J$ is indexed over all pixels of image τ, such that $\forall x_{\tau,j} = 0 \dots \texttt{width} - 1$ and $y_{\tau,j} = 0 \dots \texttt{height} - 1$. Here, $\odot$ is a Hadamard product, *i.e.* a component-wise multiplication. Thus, pixels for which $\boldsymbol{\rho}_\mathrm{mask} = 0$ do not affect the cost. Hence,

PatchMatch finds a solution to

$$\underset{\{\mathbf{p}\}_{j=1}^{J}}{\operatorname{argmin}} \sum_{j=1}^{J} \psi_{\text{color}}[\tau, \mathbf{p}_j] \; . \tag{5.7}$$

Obviously, the minimum can be found by an exhaustive search over all possible $\mathbf{p}_j$. This, clearly, is too slow for practical applications. For a fixed source image index $i_j = i_{\text{fixed}}$, $\alpha = 0$ and $s = 1$, and discrete values of x and y in $\mathbf{p}_j$, an approximate solution can be efficiently found with PatchMatch, which was proposed by Barnes *et al.* [154]. The search is done over a grid of nodes, which, in our case, correspond to the centers of the patches in the image τ. The PatchMatch algorithm iterates between two steps:

Search: at each node p_j, do a randomized search over the valid patches, *i.e.*

$$\mathbf{p}_j \leftarrow \begin{bmatrix} \text{draw } x_j \text{ from } \texttt{Uniform}[0 \ldots \texttt{width} - 1] \\ \text{draw } y_j \text{ from } \texttt{Uniform}[0 \ldots \texttt{height} - 1] \\ 0 \\ 1 \\ i_j \end{bmatrix}^T , \tag{5.8}$$

where $i_j = i_{\text{fixed}}$. Alternatively, one can search in some vicinity of the current estimate of $\mathbf{p}_j$.

Propagation: each node proposes matches to the neighboring patches of the image τ, *i.e.*

$$\mathbf{p}_j \leftarrow \mathbf{p}_{j'} + \begin{bmatrix} x_{\tau,j'} - x_{\tau,j} \\ y_{\tau,j'} - y_{\tau,j} \\ 0 \\ 0 \\ 0 \end{bmatrix}^T , \tag{5.9}$$

where j' is a set of the 4-connected neighbors of the patch centered at $[x_{\tau,j}, y_{\tau,j}, 0, 1, 0]$. This exploits the spatial smoothness of the correspondences.

Search and propagation steps are illustrated in Figure 5.6.

With a similar reasoning, generalized PatchMatch [158] extends PatchMatch to the general case of valid patches that allows sub-pixel patch coordinates, rotations and scaling (*i.e.* also searching over α and s).

PatchMatch Belief Propagation Although PatchMatch exploits the smoothness of correspondences to increase the search rate, it does not en-

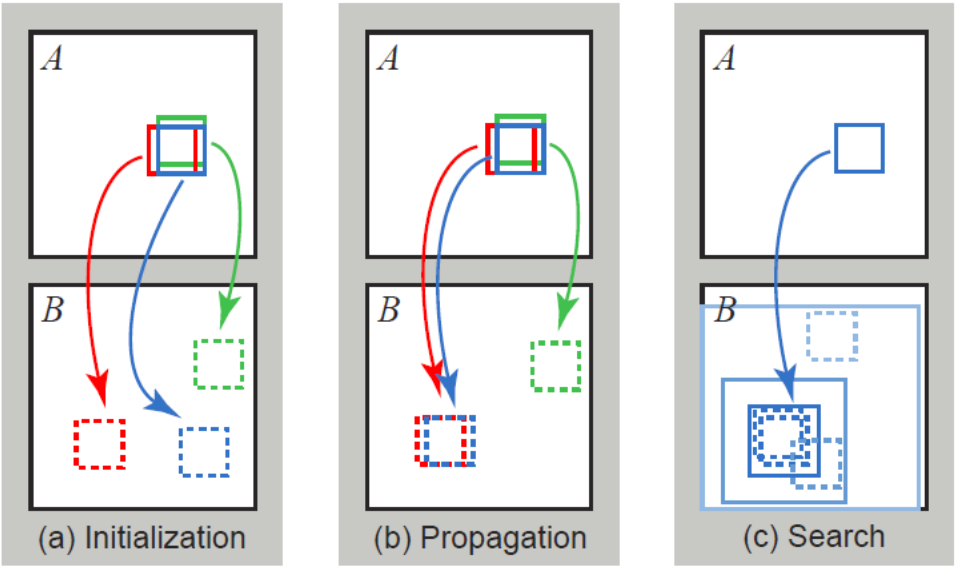


Figure 5.6: Revisiting Figure 2.10. The phases of PatchMatch algorithm: (a) patches initially have random assignments; (b) the blue patch checks above/green and left/red neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods. [Source: [154]].

courage piecewise smoothness in the estimated solution. So, it does not take into account any pairwise costs of the objective energy function, *e.g.* the cost associated with $\mathbf{p}_j$ and $\mathbf{p}_{j'}$ being dissimilar. For our purposes it is important to incorporate pairwise costs, as we want the pixels to be copied in local regions from the souce images, as the gradients of the regions contain the high-frequency details. Indeed, using PatchMatch to minimize a unary-only objective function may produce matches that are desirable in terms of the cost function, but come from difference source images. Hence, using patches that are locally and globally coherent with the blurry image, will not result in detail hallucination, but will result in reconstruction of the target image using patches of the source images.

Hence, we require a minimization of a Conditional Random Field, which consists of unary and pairwise cost terms. Belief Propagation [247, 248] is a method for finding a minimum of a Conditional Random Fields. Besse *et al.* [12] proposed to combine Particle Belief Propagation [249, 250] with PatchMatch to efficiently minimize the unary costs by sampling the state space with PatchMatch strategy. Besse *et al.* demonstrated that PatchMatch Belief Propagation is not only faster than Belief Propagation, but more accurate than PatchMatch,

due to the pairwise costs, for the problems of stereo correspondence and optical flow estimation. Next, we define some of the pairwise cost terms between a node j and its 4-neighbor j'.

Pairwise spatial smoothness cost

$$\psi_{\text{disp}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right] = \frac{(x_j - x_{j'})^2 + (y_j - y_{j'})^2}{\sqrt{\texttt{width}^2 + \texttt{height}^2}} \ . \tag{5.10}$$

Pairwise rotation smoothness cost

$$\psi_{\text{rot}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right] = (\sin[\alpha_j] - \sin[\alpha_{j'}])^2 + (\cos[\alpha_j] - \cos[\alpha_{j'}])^2 \ . \tag{5.11}$$

Pairwise scale smoothness cost

$$\psi_{\text{scale}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right] = (s_j - s_{j'})^2 \ . \tag{5.12}$$

Furthermore, we define a pairwise equality penalty

$$\psi_{\text{eq}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right] = \delta[(x_j - x_{j'})^2 + (y_j - y_{j'})^2 \leq \beta_{\text{eq}}] \ , \tag{5.13}$$

where $\delta[\cdot]$ is Kronecker's delta function which takes value 1 when the logical expression in the argument is true, and 0 otherwise. The pairwise equality penalty increases energy in the cases where the neighboring pixels of the target image are explained by patches that are too similar (almost equal), where β_{eq} controls the radius of "equality". This penalty counters the visual artifact of repeating patches.

Finally, we define a unary cost for the semantic label mismatch

$$\psi_{\text{label}}(\tau, \mathbf{p}_j) = ||\rho_{\text{label}}\left[[x_{\tau,j}, y_{\tau,j}, \tau]\right] - \rho_{\text{label}}\left[\mathbf{p}_j\right]||^2 \ . \tag{5.14}$$

We notice that the label mismatch cost compares only a single pair of pixels, *i.e.* the pixel of the target image and the central pixel of the patch.

Building Graph of Correspondences In order to allow us to search for patches across multiple source images, we build a directed graph of correspondences $\mathcal{G}$. Each patch of each image of the library corresponds to one vertex of $\mathcal{G}$. Edges of $\mathcal{G}$ only connect patches that belong to different images. For practicality, for each vertex of the graph we store a fixed number of edges that have the most similar appearance, *i.e.* the connected patches have the lowest associated costs.

The graph of correspondences $\mathcal{G}$ is used at the propagation stage of PatchMatch. So, in addition to suggesting spatial neighbors of good matches, we can suggest a sample from the connected patches in $\mathcal{G}$ from the vertex associated with the current correspondence

$$\mathbf{p}_j \leftarrow \mathbf{p}_{j\dagger} + \begin{bmatrix} 0 \\ 0 \\ \alpha_j \\ s_j - s_{j\dagger} \\ 0 \end{bmatrix}^T , \tag{5.15}$$

where $\mathbf{p}_{j\dagger}$ is randomly chosen correspondence that is connected to $\mathbf{p}_j$ in $\mathcal{G}$.

To build the graph, we first estimate the best correspondences for all patches between all pairs of images. So, for each pair of images $\tau = 1 \dots I$ and $i = 1 \dots I, i \neq \tau$, we precompute the approximate minumum of the objective function

$$\begin{aligned} \mathcal{L}_{\text{graph}}[\tau, i] = &\sum_{j=1}^{J} \left(\psi_{\text{color}} \left[\tau, \mathbf{p}_j \right] + \lambda_{\text{label}} \psi_{\text{label}} \left[\tau, \mathbf{p}_j \right] \right) \\ &+ \frac{1}{2} \sum_{j'} \left(\lambda_{\text{disp}} \psi_{\text{disp}} \left[\mathbf{p}_j, \mathbf{p}_{j'} \right] + \lambda_{\text{rot}} \psi_{\text{rot}} \left[\mathbf{p}_j, \mathbf{p}_{j'} \right] \right) , \end{aligned} \tag{5.16}$$

where τ is the target image i is the source image and $\lambda_{\cdot}$ are scalar weights of respective terms. We restrict the patch coordinates $\mathbf{p}_j$ to have $\alpha_j \in \{0, 45, 90, 135, 180, 225, 270, 315\}^\circ$, $s_j = 1$ and $i_j = i$.

Next, for every patch $\mathbf{p}_j$ of every image $\tau = 1 \dots I$, we store the best $N_{\mathcal{G}}$ matches among all solutions of $L_{\text{graph}}[\tau, i]$, where $i = 1 \dots I, i \neq \tau$. These best $N_{\mathcal{G}}$ matches are used as edges in $\mathcal{G}$, and they are uniformly sampled as suggestions at the propagation stage, as shown in equation 5.15.

The use of $\mathcal{G}$ significantly improves the convergence of PatchMatch Belief Propagation. To demonstrate this, we run PatchMatch Belief Propagation algorithm with and without the correspondence suggestions from the graph of correspondences $\mathcal{G}$. For fairness, we propagate an additional random correspondence as a candidate when running without the use of $\mathcal{G}$. As the objective, we use

$$\mathfrak{L}[\tau, i] = \sum_{j=1}^{J} \left(\psi_{\text{color}} \left[\tau, \mathbf{p}_j \right] + \lambda_{\text{label}} \psi_{\text{label}} \left[\tau, \mathbf{p}_j \right] \right) . \tag{5.17}$$

We measure the energy for each image of the test set as τ. Figure 5.7 shows the mean energy gain ($\mathfrak{L}_{\text{without } \mathcal{G}}/\mathfrak{L}_{\text{with } \mathcal{G}}$) of using the graph of correspondences $\mathcal{G}$ per iteration of PMBP. Hence, before any iterations of PMBP, the ratio of energies is at 1.0. However, after only 5 iterations of PMBP with the graph of correspondences, the energy is significantly lower than without the use of the graph of correspondences.

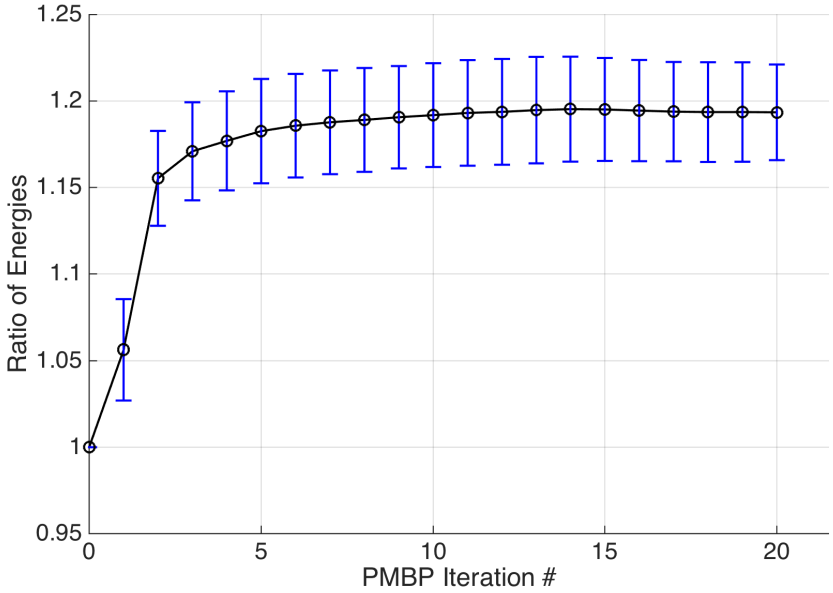


Figure 5.7: The mean energy gain ($\mathfrak{L}_{\text{without } \mathcal{G}}/\mathfrak{L}_{\text{with } \mathcal{G}}$, see equation 5.17) of using the graph of correspondences $\mathcal{G}$ per iteration of PatchMatch Belief Propagation. (Higher is better.)

5.3.2 Detail Hallucination of C-CCA Sample

In this subsection, we focus on detail hallucination of image T. First, we search for suitable patches of the image T by minimizing an objective function

$$\begin{aligned}\mathcal{L}[T] = &\sum_{j=1}^{J} \left(\psi_{\text{color}}\left[T, \mathbf{p}_j\right] + \lambda_{\text{label}}\psi_{\text{label}}\left[T, \mathbf{p}_j\right]\right) \\ &+\frac{1}{2}\sum_{j'}\Big(\delta[i_j \neq i_{j'}]\lambda_{\text{i}} \\ &\quad +\delta[i_j = i_{j'}]\big(\lambda_{\text{disp}}\psi_{\text{disp}}\left[\mathbf{p}_j, \mathbf{p_{j'}}\right] + \lambda_{\text{rot}}\psi_{\text{rot}}\left[\mathbf{p}_j, \mathbf{p_{j'}}\right] \\ &\quad\quad + \lambda_{\text{scale}}\psi_{\text{scale}}\left[\mathbf{p}_j, \mathbf{p_{j'}}\right] + \lambda_{\text{eq}}\psi_{\text{eq}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right]\big)\Big)\,, \end{aligned} \tag{5.18}$$

where λ_i is the penalty cost for copying patches from different source images. So, the solution either pays the penalty λ_{i} for copying from different sources, or ensures that the pixels from the same source are adjacent to each other. We restrict the patch coordinates $\mathbf{p}_j$ to have $\alpha_j \in \{0, 45, 90, 135, 180, 225, 270, 315\}^\circ$, and $s_j \in (0.85, 1.15)$.

After a number of iterations of the PatchMatch Belief Propagation algorithm, for every pixel of image T we have a patch correspondence $\mathbf{p}_j$. Figures 5.8 (a) and 5.8 (b) visualize the patches found.

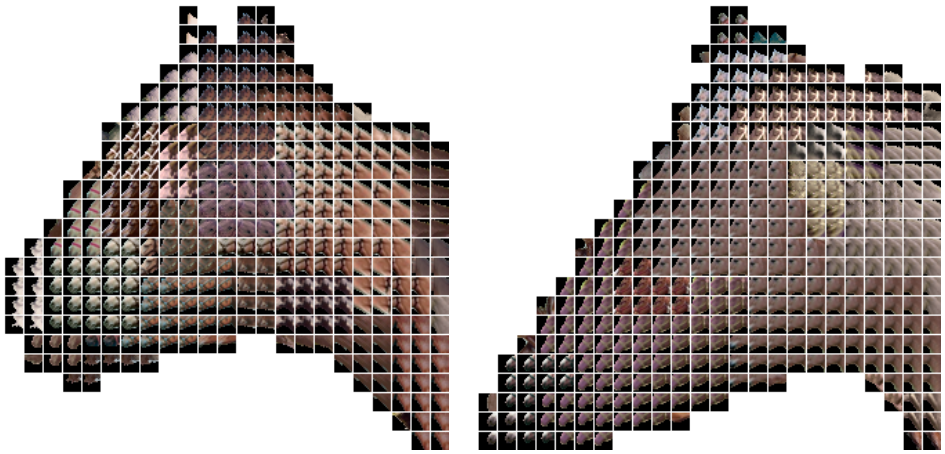
One approach of combining the found patches to generate a new image is to average the RGB values of overlapping pixels of the found patches. Figure 5.9 shows results of the blending (averaging) of the found patches. As it can be seen, some patches are not locally coherent due to color mismatch, this produces color bleeding in blending of small patches. The color mismatch effect can be reduced by preprocessing the patches to match the colors of the target image T. Figure 5.10 shows similar blending, but each patch is preprocessed before the blending, such that the mean color value of the patch is matched to the mean color value of the target patch in image T. As it is shown in Figures 5.9 and 5.10 the averaging operation reduces the high-frequency details. Although blending of smaller patches produces sharper images, it also introduces noise. Furthermore, small patches do not contain large, relevant, features of the visual object, for example the eyes of the horse.

Another approach is to blend the patches in the gradient domain, *i.e.* Poisson Blending [203], as shown in Figure 5.11. However, for small patch sizes this results in color bleeding. For larger patches, the small detailes are "averaged away" as the gradient at each pixel is an average of 49 or more (the size of the used patch, *e.g.*, $7x7$) values. Notice, that mean color matching operation doesn't effect the blending in the gradient domain. Nevertheless, the quality of the blending is similar to pixel averaging.

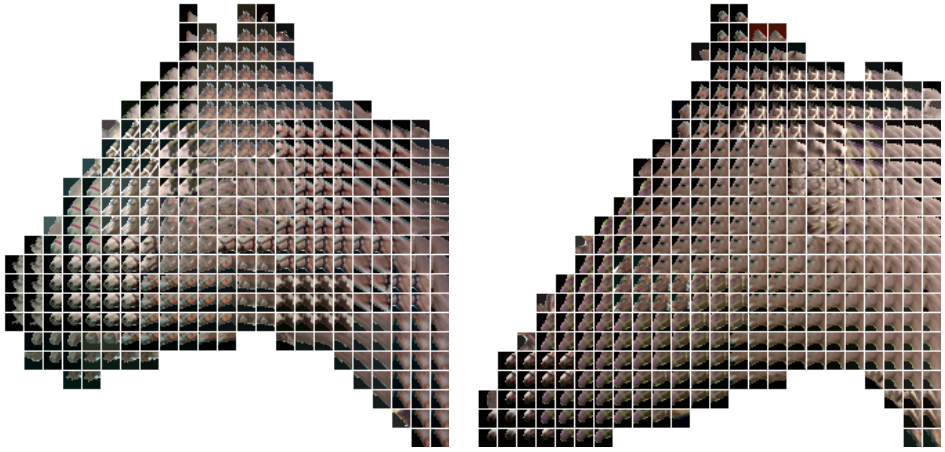
Detailed Partial Image Our approach of including details contained in the patches found by PMBP algorithm (as can be seen in Figure 5.9) is to generate an interim, detailed, but incomplete, image T_{partial} that is used as an additional term in another iteration of PMBP patch search.

Hence, we generate the detailed partial image by choosing non-overlapping, circular patches that have a high variance of color values. The patches are chosen sequentially, from the highest variance to the lowest variance, such that each patch is locally coherent (at least 7 of the 8-connected neighbour nodes point to the same image index) and it is not too close to the previously chosen

(a) Target image T.

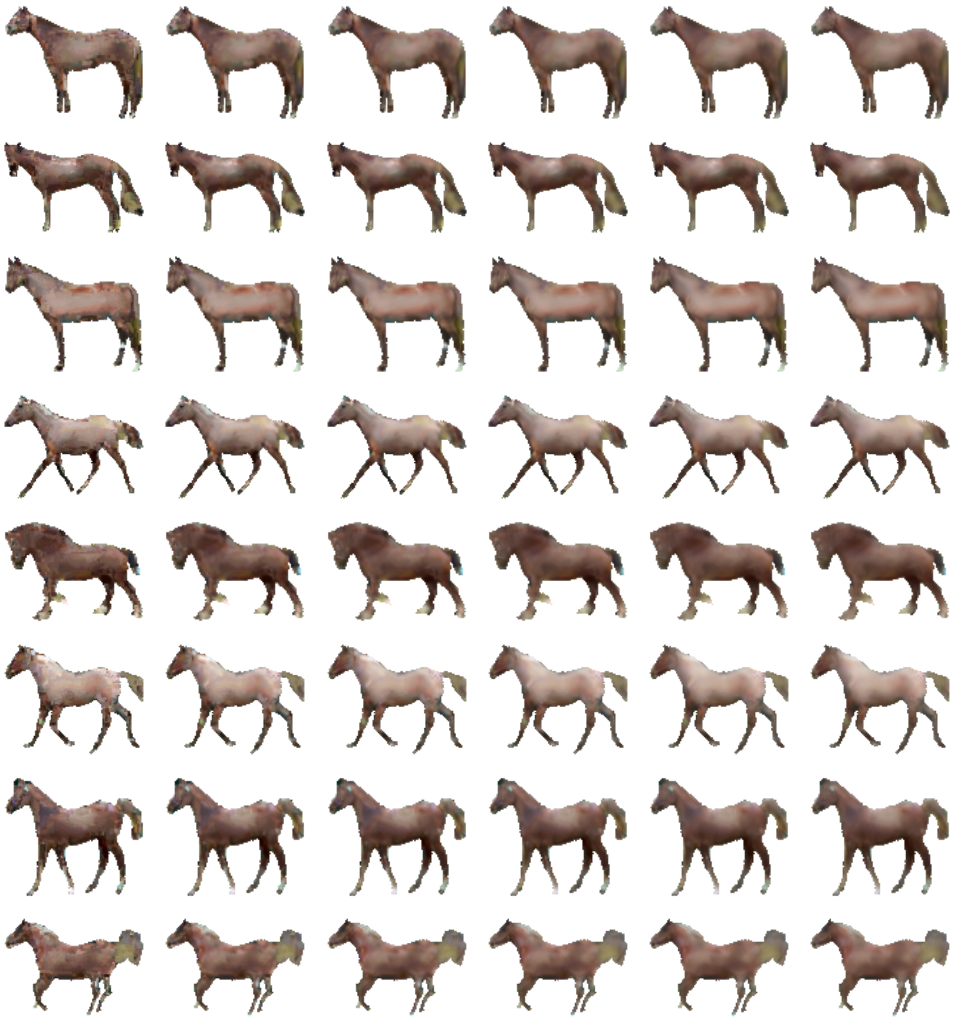


(b) Patches corresponding to patches centered at each pixel in the red window of (a)



(c) Similar to (b), but for each patches the mean value is matched to the mean value of the target patch in T.

Figure 5.8: Patches found by PMBP algorithm.



(a) 3×3 patches | (b) 5×5 patches | (c) 7×7 patches | (d) 9×9 patches | (e) 11×11 patches | (f) 13×13 patches

Figure 5.9: Blending of patches found by PMBP algorithm for the target images in Figure 5.3 (the first random draw). The patches have sizes from 3×3 to 13×13 as shown in (a) to (f), respectively. The patches are averaged with the Gaussian weighting, where the standard deviation of the Gaussian is the quarter of the size of the patch.

patches (there is a gap of at least 2 pixels between any two chosen patches). Figure 5.12 show examples of the detailed partial image generated with different patch sizes. The generated images are incomplete when patches from some region do not pass the local coherence constraint. As it can be seen, a lot of patches contain details that are desirable in the final result.

The detailed partial image T_{partial} is used in the additional unary term for

(a) 3×3 patches | (b) 5×5 patches | (c) 7×7 patches | (d) 9×9 patches | (e) 11×11 patches | (f) 13×13 patches

Figure 5.10: Blending of patches found by PMBP algorithm for the target images in Figure 5.3 (the first random draw). The patches have sizes from 3×3 to 13×13 as shown in (a) to (f), respectively. The mean of each corresponding patch is matched to the mean of the patch in the target image. The patches are averaged with the Gaussian weighting, where the standard deviation of the Gaussian is the quarter of the size of the patch.

the objective function

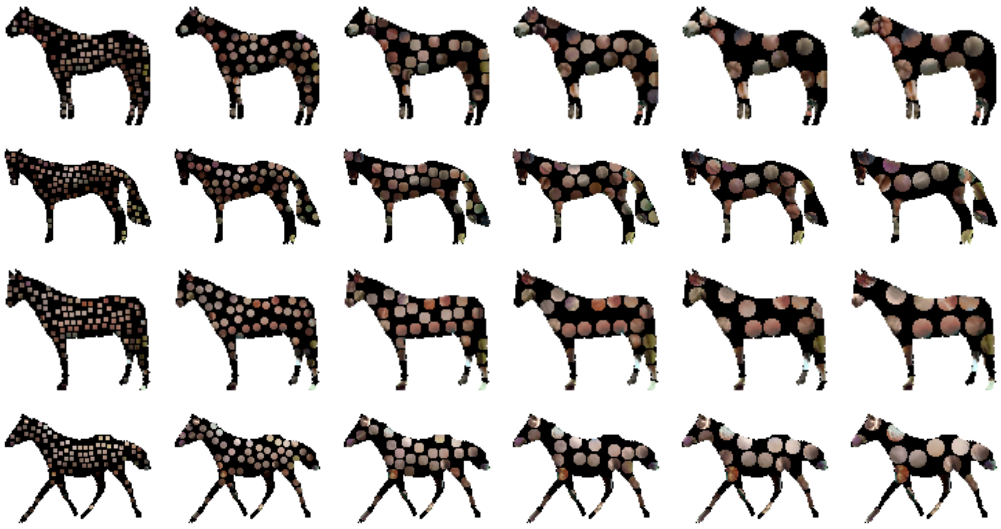
$$\psi_{\text{partial}}[T_{\text{partial}}, \mathbf{p}_j] = ||\boldsymbol{\rho}_{\text{mask}}\left([x_{\tau,j}, y_{\tau,j}, 0, 1, T_{\text{partial}}]\right) \odot \left(\boldsymbol{\rho}_{\text{color}}\left([x_{\tau,j}, y_{\tau,j}, 0, 1, T_{\text{partial}}]\right) - \boldsymbol{\rho}_{\text{color}}\left[\mathbf{p}_j\right]\right)||^2 . \quad (5.19)$$

(a) 3×3 patches | (b) 5×5 patches | (c) 7×7 patches | (d) 9×9 patches | (e) 11×11 patches | (f) 13×13 patches

Figure 5.11: Poisson blending of patches found by PMBP algorithm for the target images in Figure 5.3 (the first random draw). The patches have sizes from 3×3 to 13×13 as shown in (a) to (f), respectively. The horizontal and vertical gradients of patches are averaged with the Gaussian weighting, where the standard deviation of the Gaussian is the quarter of the size of the patch. The computed horizontal and vertical gradient fields are used in the Poisson equation [203], with the boundary gradient constrained to equal to the boundary gradient of the target image T.

Hence, similarly to equation 5.18, we define a second objective function

$$\begin{aligned}\mathcal{L}[T, T_{\text{partial}}] = &\sum_{j=1}^{J} \left(\psi_{\text{color}}\left[T, \mathbf{p}_j\right] + \lambda_{\text{label}}\psi_{\text{label}}\left[T, \mathbf{p}_j\right] + \lambda_{\text{partial}}\psi_{\text{partial}}[T_{\text{partial}}, \mathbf{p}_j]\right) \\ &+\frac{1}{2}\sum_{j'} \Big(\lambda_{\text{i}}\delta[i_j \neq i_{j'}] + \delta[i_j = i_{j'}]\big(\lambda_{\text{disp}}\psi_{\text{disp}}\left[\mathbf{p}_j, \mathbf{p}_{\mathbf{j'}}\right] + \lambda_{\text{rot}}\psi_{\text{rot}}\left[\mathbf{p}_j, \mathbf{p}_{\mathbf{j'}}\right] \\ &+\lambda_{\text{scale}}\psi_{\text{scale}}\left[\mathbf{p}_j, \mathbf{p}_{\mathbf{j'}}\right] + \lambda_{\text{eq}}\psi_{\text{eq}}\left[\mathbf{p}_j, \mathbf{p}_{j'}\right]\big)\Big)\,, \end{aligned} \tag{5.20}$$



(a) 3×3 patches | (b) 5×5 patches | (c) 7×7 patches | (d) 9×9 patches | (e) 11×11 patches | (f) 13×13 patches

Figure 5.12: Detailed partial images generated with different patch sizes.

which is minimized with PMBP.

Figure 5.13 shows the blendings of the patches found by minimizing $\mathcal{L}[T]$ and $\mathcal{L}[T, T_{\text{partial}}]$. Notice, the change in image indices of the found patch correspondences: image indices in Figure 5.13(d) have "smoother" image index regions.

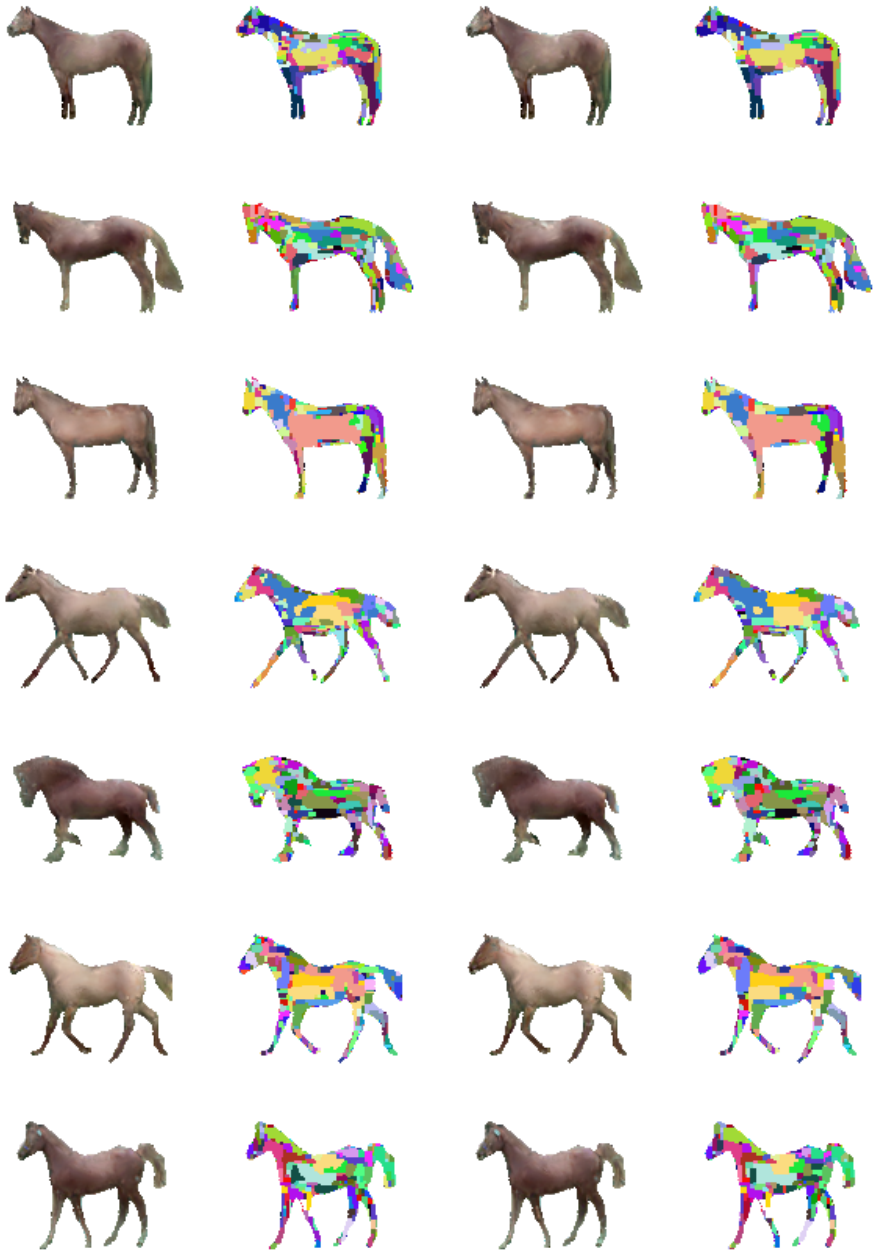
Colorspace Transformation Finally, we transform the colorspace of T_{detailed} with random draws of $\mathbf{R}_\star$ and $\mathbf{t}_\star$ colorspace parameters with the sampling method discussed in section 5.2.2.

5.4 Results

The hyperparameters of PatchMatch Belief Propagation are chosen empirically and are shown in Table 5.1.

Figures 5.16 and 5.19 show results of our model applied to Horses dataset. The C-CCA model was trained on 250 images. The segmentations and part labels of 45 images of the test set (see Figure 5.14) were used to draw samples of C-CCA (see Figures 5.15 and 5.18). Patches of size 9×9 were used to generate the detailed partial images T_{partial} (see Figure D.1). For the final result in Figure 5.16, patches of size 3×3 were averaged for all patch correspondences.

Figures 5.17 and 5.20 visualizes image indices of the estimated patch correspondences. This illustrates that the each of the generated images consist



(a) (b) (c) (d)

Figure 5.13: (a) Blending of 3×3 patches found by minimizing $\mathcal{L}[T]$. (b) Visualization of image indices of patches used to generate results in (a). (c) Blending of 3×3 patches found by minimizing $\mathcal{L}[T, T_{\text{partial}}]$. (d) Visualization of image indices of patches used to generate results in (c).

of fragments of other images and the results is not a warped version of only one images of the training set.

Our local non-parametric model has improved the visual quality of the samples of C-CCA (see Figures 5.21, 5.22, 5.33 and 5.34 for side-by-side

Parameters	$\mathcal{L}_{\text{graph}}[\tau, i]$	$\mathcal{L}[T]$	$\mathcal{L}[T, T_{\text{partial}}]$
n_{p}	13	13	13
λ_{label}	3	2	3
λ_{disp}	0.3	0.2	0.2
λ_{rot}	0.3	0.02	0.02
λ_{scale}	n/a	0.02	0.02
λ_{i}	n/a	1.5	1.5
λ_{eq}	n/a	1	1
λ_{partial}	n/a	n/a	6

Table 5.1: PMBP parameters used in our experiments.

comparison); image features such as eyes, muscles and mane were introduced to some of the samples of C-CCA.

The training set for Elephants dataset consists of 250 images and test set consists of 25 images (see Figure 5.23). Similarly, Figures 5.25, 5.28 and 5.31 shows results for Elephants dataset and Figures 5.26, 5.29 and 5.32 visualize image indices of the estimated patch correspondences, respectively.

Unlike the horses dataset, the images of elephants have in-plane rotation of the elephants with respect to the camera. Hence, learning a parametric model for the elephants is more difficult. Consequently, samples of elephants are of poor quality compared to horses, which, leads to poor quaity of our final results. Nevertheless, our local model has improved the quality of C-CCA samples: image features such as wrinkles on the trunk and eyes were introduced to some of the target images.

Figure 5.14: Test set images of the horses dataset.

Figure 5.15: Images of horses in the common colorspace synthesized with a random sample of component weights $\mathbf{h}_\star$ and blurred with a Gaussian kernel. Each image corresponds to image T used to generate results in Figure 5.16.

Figure 5.16: Final synthesis results for the horses dataset with a random sample of parameters $\mathbf{h}_\star$, $R_\star$ and $\mathbf{t}_\star$.

Figure 5.17: Visualization of image indices of the patches used to generate results in Figure 5.16.

Figure 5.18: Images of horses in the common colorspace synthesized with a random sample of component weights $\mathbf{h}_\star$ and blurred with a Gaussian kernel. Each image corresponds to image T used to generate results in Figure 5.19.

Figure 5.19: Final synthesis results for the horses dataset with another sample of parameters $\mathbf{h}_\star$, $R_\star$ and $\mathbf{t}_\star$.

Figure 5.20: Visualization of image indices of the patches used to generate results in Figure 5.19.

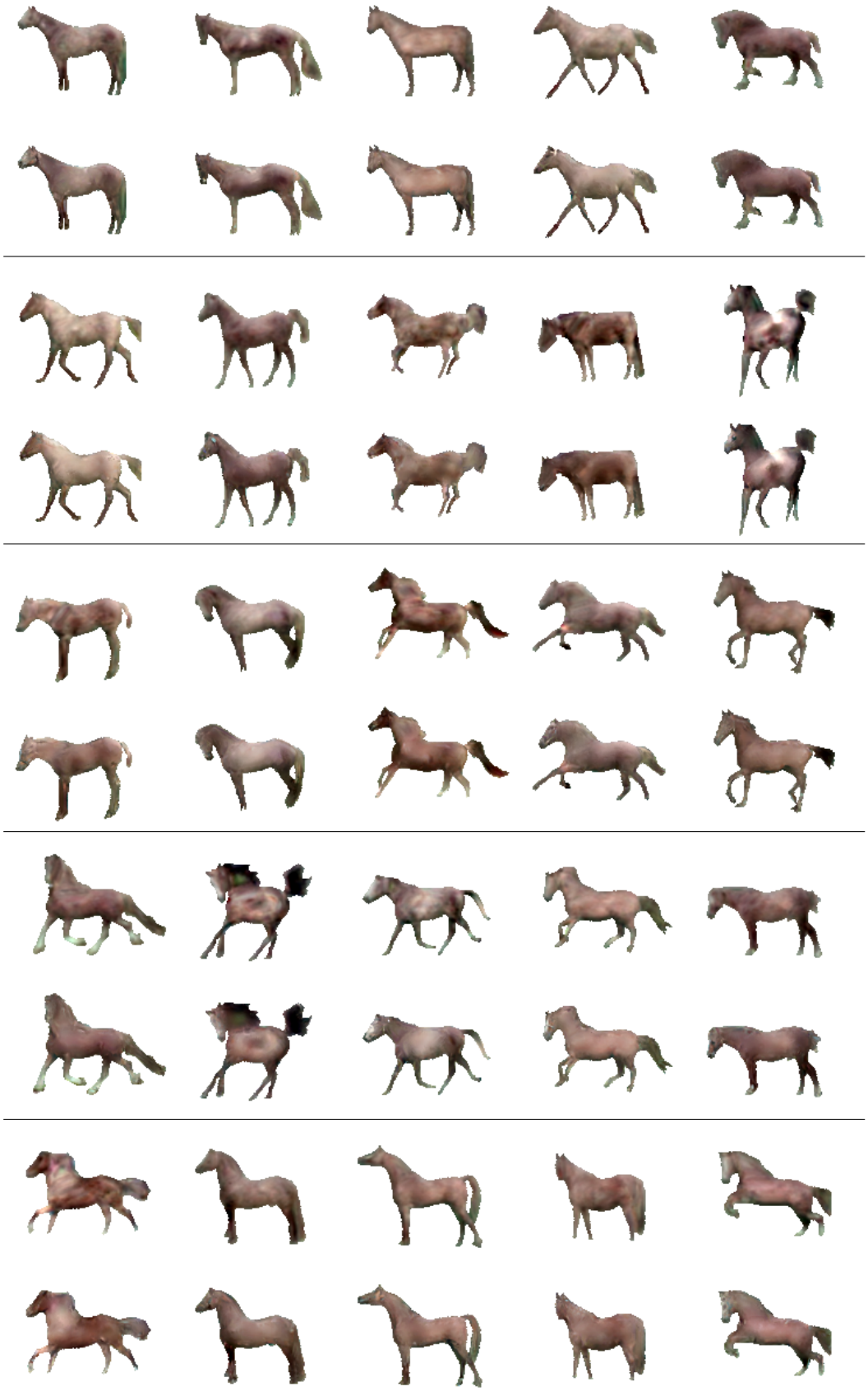


Figure 5.21: Each odd row is an image of a horse synthesized with C-CCA from Figure 5.16. Each even row is an image of a horse after detail hallucination.

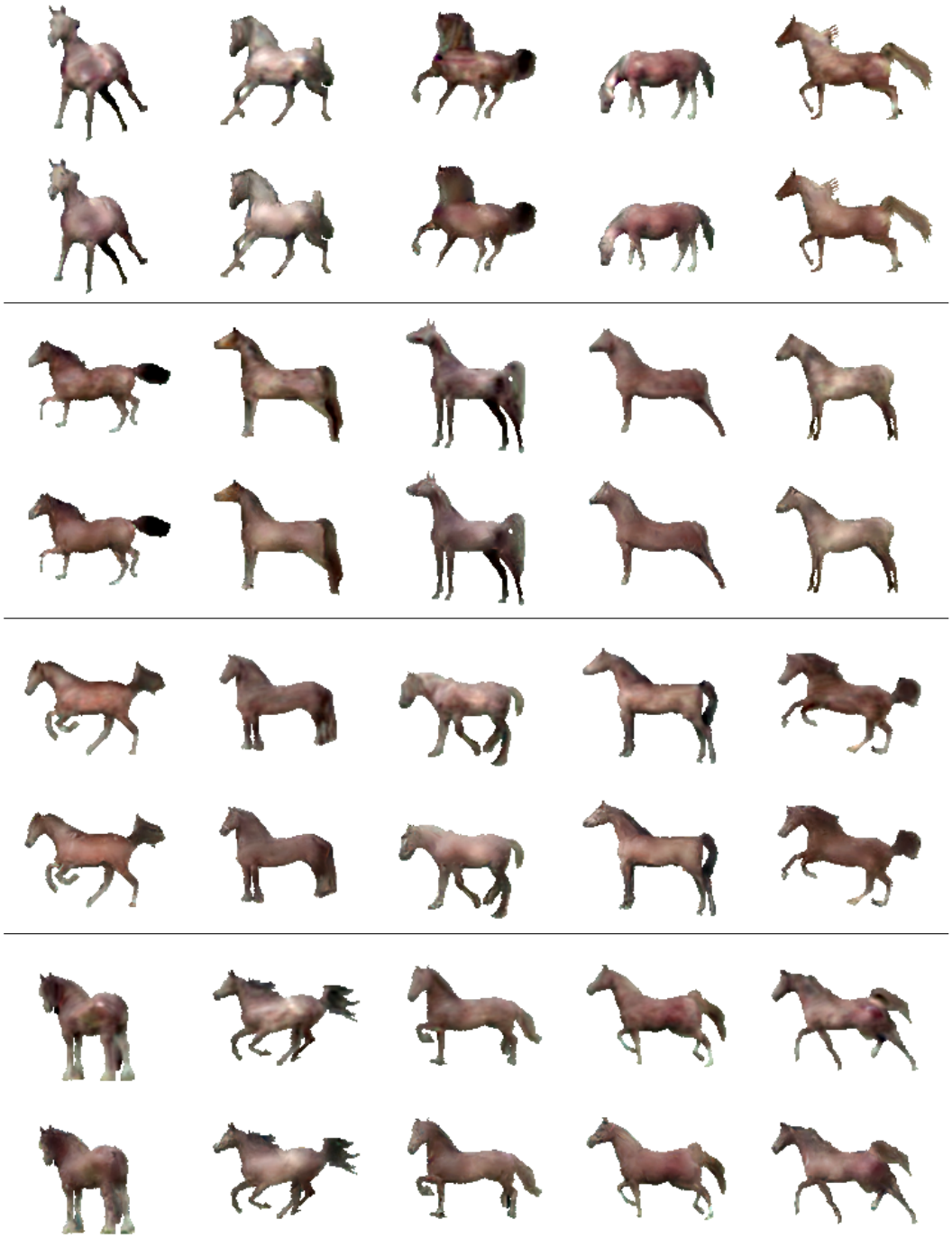


Figure 5.22: Continued from Figure 5.21: Each odd row is an image of a horse synthesized with C-CCA from Figure 5.16. Each even row is an image of a horse after detail hallucination.

Figure 5.23: Test set images of the elephants dataset.

Figure 5.24: Images of elephants in the common colorspace synthesized with a random sample of component weights $\mathbf{h}_\star$ and blurred with a Gaussian kernel. Each image corresponds to image T used to generate results in Figure 5.25.

Figure 5.25: Final synthesis results for the elephants dataset with a random sample of parameters $\mathbf{h}_\star$, $R_\star$ and $\mathbf{t}_\star$.

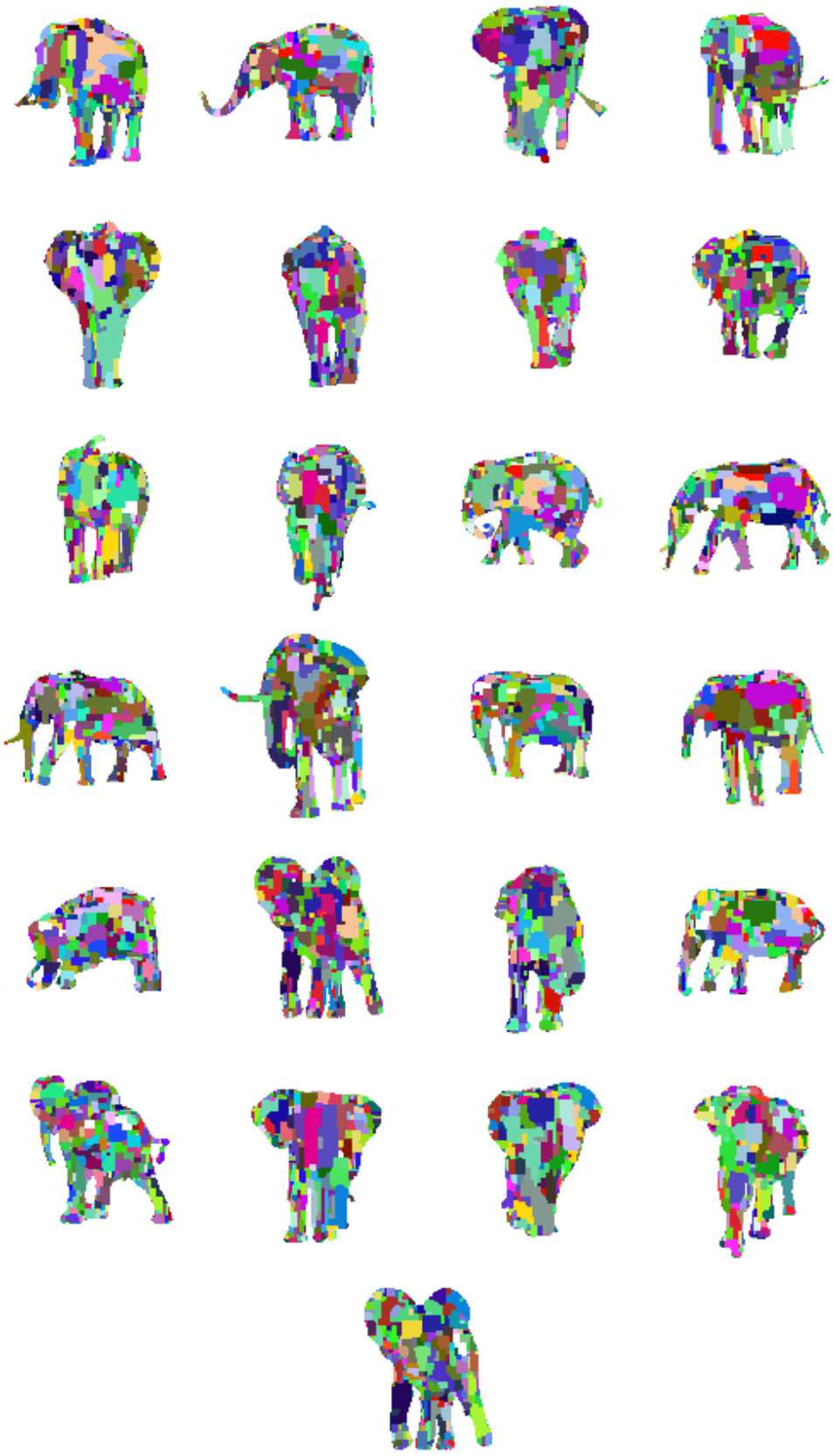


Figure 5.26: Visualization of image indices of the patches used to generate results in Figure 5.25.

Figure 5.27: Images of elephants in the common colorspace synthesized with another random sample of component weights $\mathbf{h}_\star$ and blurred with a Gaussian kernel. Each image corresponds to image T used to generate results in Figure 5.28.

Figure 5.28: Final synthesis results for the elephants dataset with another sample of parameters $\mathbf{h}_\star$, $R_\star$ and $\mathbf{t}_\star$.

Figure 5.29: Visualization of image indices of the patches used to generate results in Figure 5.28.

Figure 5.30: Images of elephants in the common colorspace synthesized with another random sample of component weights $\mathbf{h}_\star$ and blurred with a Gaussian kernel. Each image corresponds to image T used to generate results in Figure 5.31.

Figure 5.31: Final synthesis results for the elephants dataset with another sample of parameters $\mathbf{h}_\star$, $R_\star$ and $\mathbf{t}_\star$.

Figure 5.32: Visualization of image indices of the patches used to generate results in Figure 5.31.

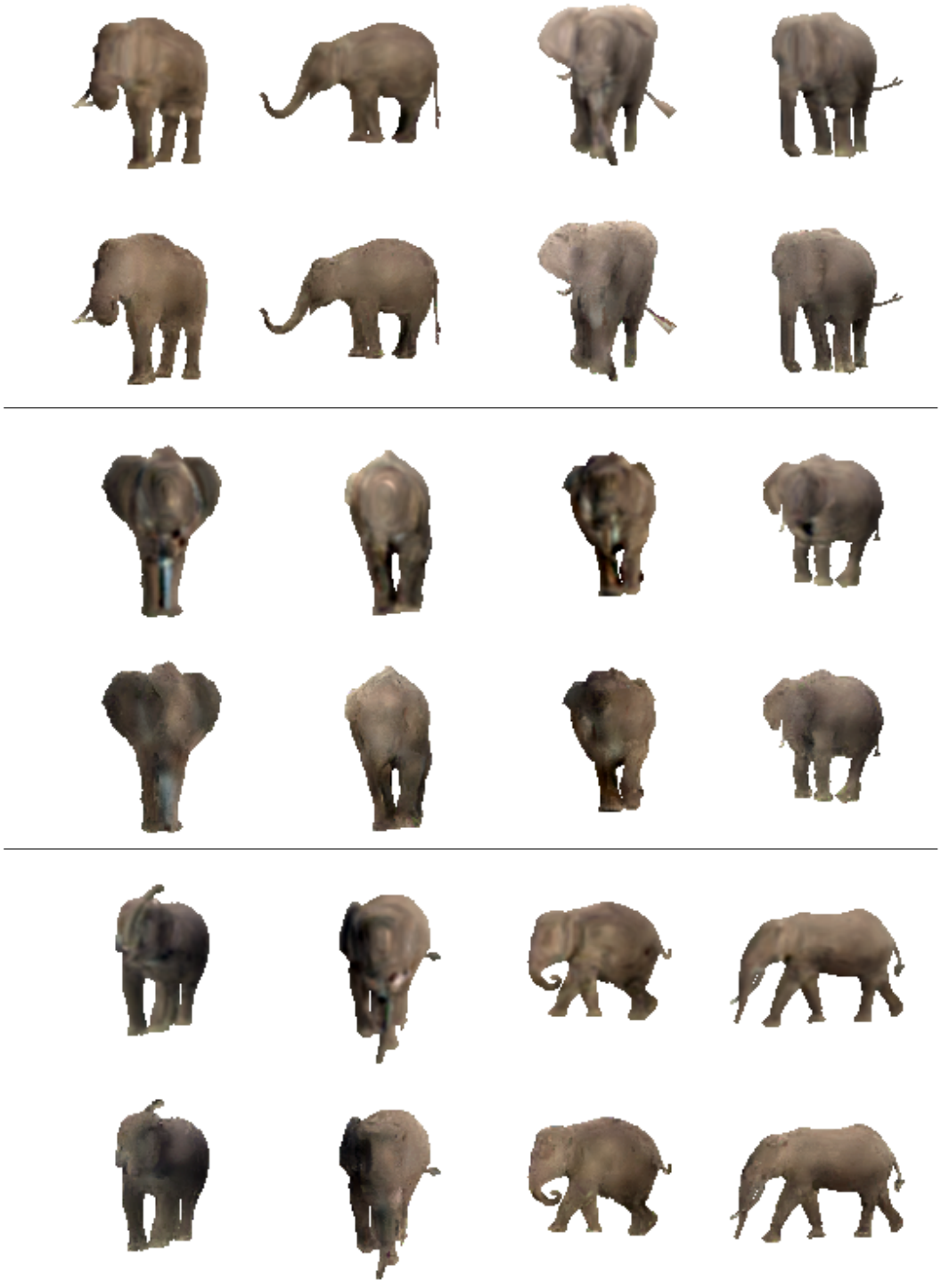


Figure 5.33: Each odd row is an image of an elephant synthesized with C-CCA from Figure 5.24. Each even row is an image of an elephant after detail hallucination.

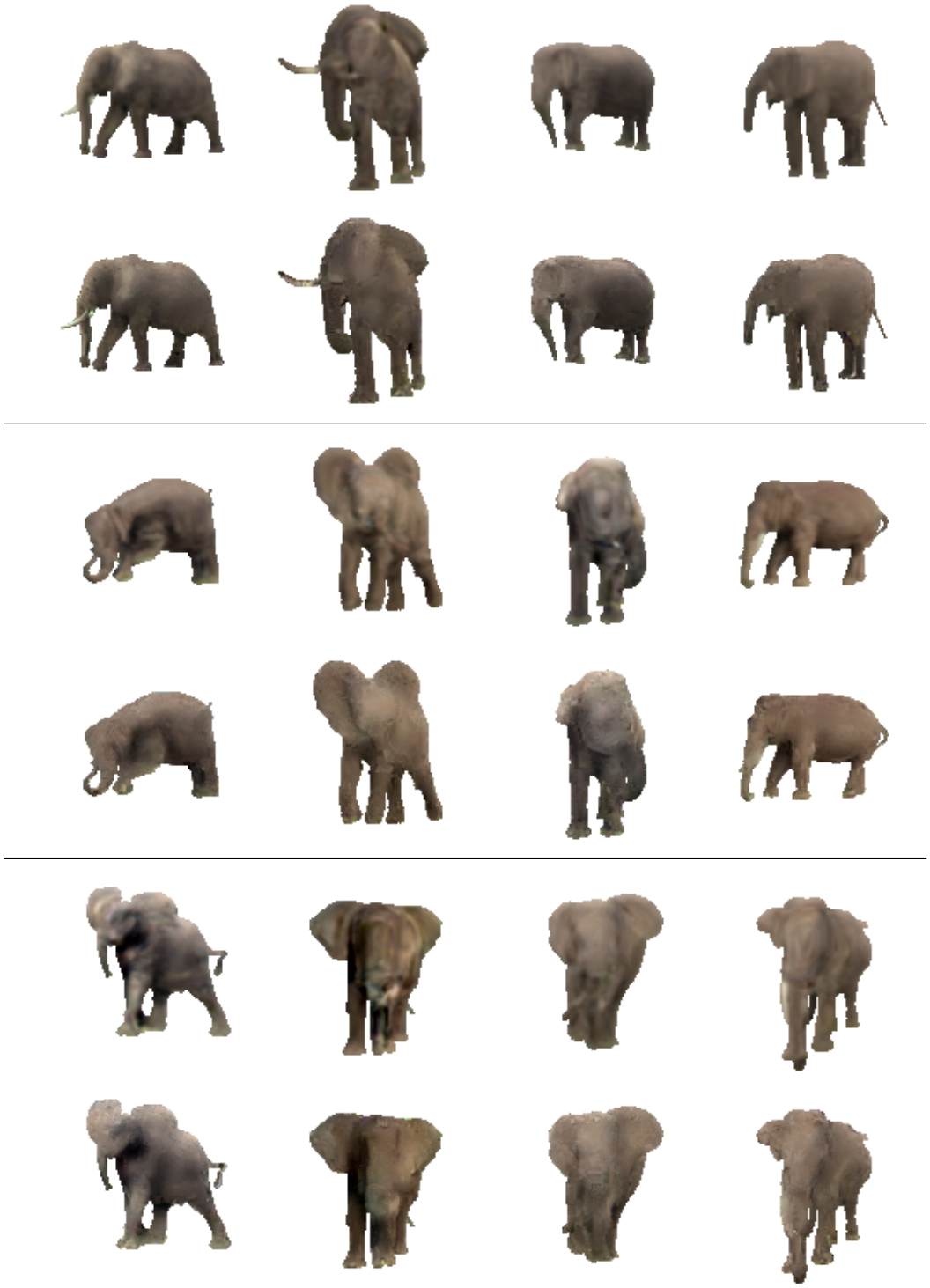


Figure 5.34: Continued from Figure 5.33. Each odd row is an image of an elephant synthesized with C-CCA from Figure 5.24. Each even row is an image of an elephant after detail hallucination.

5.5 Conclusion

In this chapter we investigated the challenging problem of automatic image synthesis of highly-deformable visual objects, specifically horses and elephants, from a dataset of images. Our approach combines the globally-coherent parametric model and locally-coherent non-parametric model in a two-stage pipeline.

We demonstrate how to draw a sample from Context-Conditioned Component Analysis, trained on a dataset of images with corresponding segmentations and part-labels, to generate a globally-coherent novel image T. We explore the latent space of appearances learned with C-CCA and demonstrate that the appearance variations along different dimensions of the latent space are consistent, even under different poses of the visual objects.

The lack of high-frequency details in the generate image T is addressed with the second stage. We developed a non-parametric model that works on a local patch level. The non-parametric model finds patches in the dataset that correspond to patches in the image T. The state-of-the-art PatchMatch Belief Propagation algorithm [12] was used to efficiently solve the patch correspondence problem. We precomputed a graph of inter-image correspondences [230] to increase search speed. The found patches were blended to generate an image with hallucinated details. Finally, colorspace transformations were applied to synthesize a final result.

We demonstrated synthesis results of our method on datasets of horses and elephants.

5.6 Future Work

In this work, after the non-parametric model, the resolution of generated images remains the same. However, for future work, higher-resolution patches from the dataset could be used to both hallucinate high-frequency details and increase the spatial resolution of the synthesized images.

The patches are compared in the "common" RGB colorspace, but comparison in other colorspaces, such as *Lab*, may improve the results. For any colorspace representations, the patches of the dataset and image T could be normalized before comparison to introduce invariance to the global intensity of the object and increase the pool of suitable patches. Another alternative is to compare patches in some feature space, rather than color space. The feature space could be filter responses to Gabor-like filterbank [1], SIFT desciptors [175], Neural Network feature activations [231], *etc*. So, the investigation of other

cost functions for the patch correspondence problem is required.

Finally, our local model uses C-CCA output as input, hence, the quality of the final result depends on the quality of the samples of C-CCA. As C-CCA was not trained to generate good samples, but rather trained to reconstruct training images, there is no guarantee that all random samples of C-CCA are realistic. Optimizing another loss function that explicitly models the realism of random samples, such as adversarial loss [144], is left as future work. Furthermore, a richer set of part labels should improve the samples of C-CCA, which in turn would increase the quality of the synthesized images.

Chapter 6

Conclusions

In this thesis we investigated the problem of image synthesis. In particular we explored non-parametric methods, parametric methods and combinations of the two for generating images, with the focus on synthesis of deformable visual objects. Our approach exploits a structure map: an image of discrete labels where each label indicates the part of the object that is present at that position.

We first reviewed literature on the problem of image synthesis in Chapter 2 and concluded that existing methods synthesize high-quality results for visual objects that are aligned and have fixed structure, but they are not readily applicable for more complex, deformable visual objects.

In Chapter 3 we investigated the problem of interactive image synthesis of deformable visual objects. We collected datasets of various animals with corresponding segmentations and semantic label maps. Based on traditional illustrator's workflow, we represented the animal's pose using a set of ellipses as a proxy to allow the users an easy and intuitive way of specifying the pose of the animal. We modeled the joint representation of the pose and contour in a GP-LVM, which allowed the system to synthesize continuous suggestions to the user's input. The system allowed the user to select a few exemplar images from the dataset based on the pose and color similarity. Features extracted from the user's sketch were used to successfully guide the non-parametric model that synthesized novel images of non-aligned, deformable visual objects from the selected exemplars. The global consistency was enforced by sparseness of the selected exemplars. The resulting images were generated in a coarse-to-fine fashion with a non-parametric method from the selected exemplars. We evaluated the method and the synthesized images through two user studies. The results indicated that non-parametric approaches can be used to synthesize

novel images of highly deformable visual objects with the guidance of a structure map. Furthermore, the traditional illustrator's approach of specifying the gross pose of an object with masses proved to be especially useful for specifying the 3D pose.

Synthesizing novel images from a sparse set of weakly-aligned exemplars is suboptimal, as the appearances in the sparse set may be incompatible. This motivated the approach of a generative parametric model that can capture the appearance of highly-deformable or non-structured visual objects from a few hundred images and corresponding structure maps. Hence, we introduced a new generative parametric model in Chapter 4 called Context-Conditioned Component Analysis. C-CCA represents images as a linear combination of appearance functions, which in turn are conditioned on the non-appearance information such as keypoint distances, segmentation masks, *etc*. Instead of conditioning the component value at each pixel with its (x, y) image coordinate, as in PPCA, we can condition the component value on the local structural context. We showed that C-CCA is a generalization of PPCA and Active Appearance Models, but more powerful, as the local context from the structure maps can be described in several ways. We derived an efficient learning procedure for C-CCA and learned appearance representations of deformable and non-structured visual objects such as horses, elephants, cats and facades. We showed how C-CCA can estimate the appearance parameters from a partial view of the visual object in the structured image inpainting task. Furthermore, we showed how the component functions share the respective appearance variations across poses, which allowed succesful appearance transfer across different poses of the visual object. Finally, we evaluated C-CCA with quantitative and perceptual metrics for future comparisons.

For completely automatic synthesis of images, we can draw random samples from C-CCA with random parameters. Although most of the images drawn from C-CCA were globally coherent, they lacked high-frequency details. This is due to coarseness and misalignment in the labeling of the training images and dimensionality reduction applied to avoid overfitting. In Chapter 5 we showed how a combination of the parametric and non-parametric methods can be used to synthesize novel images with both global and local coherence. We used C-CCA as the parametric model to generate globally-coherent, albeit blurry, images. To improve the visual fidelity of the images, we applied non-parametric model that finds and blends patches that are close to the sampled image. The colorspace of the generated images was transformed with random samples of the

colorspace parameters to generate the final results. The non-parametric model improved the quality of the results by hallucinating high-frequency details. We showed how the synthesized image were novel, as they consisted of regions of multiple different images of the training set.

6.1 Limitations and Future Work

Labels The presented methods of image synthesis exploit semantic labels provided both at training and test times. Although the semantic labels could be provided by the user, similarly to the system described in Chapter 3, an automatic synthesis of novel, structurally-valid semantic label maps remains a future work.

Currently the set of labels has to be consistent among the images (*e.g.*, does the animal have a single left front leg label, or does it have 3 labels: an upper left leg, lower left leg and left foot?). However, a hierarchy of part labels (used for object detection) could be more intuitive to the user to constrain the synthesized image. Hence, adapting models to the hierarchical label map is left as future work.

An interesting line of research would be learning the semantic labels from the data in a semi-supervised fashion, either by reconstructing and fitting a template 3D mesh for structured visual objects (see section 2.1.2); or learning the common elements of the object for non-structured visual objects (see section 2.1.3). Morever, a greater set of part labels and keypoints should improve our parametric model, described in Chapter 4, due to better alignment of data in the context space, leading to a more accurate model.

In addition to the local semantic labels, global semantic labels could be used to condition C-CCA. This could allow further semantic control of the samples, for example, conditioning C-CCA to generate a facade of a building using the city as the global constraint (*e.g.*, "I want a facade of a building in Tokyo").

Models One could explore more powerful function approximators that could be used for $\phi[\cdot, \cdot]$ in C-CCA. For example, a Convolutional Neural Network that outputs the values of the basis functions given a semantic label map as input. Neural Networks as function approximators would also benefit from a larger training datasets.

Another line of research would be to extend C-CCA from the assumption of standard Gaussian distribution of the latent representations to more complicated, non-unimodal distributions of the latent representations.

Random samples of C-CCA do not necessarily produce realistic images as the underlying distribution of visual objects may not be Gaussian, or unimodal. Recently proposed loss functions and training methods (*e.g.*, Generative Adversarial Networks *et al.* [144, 145]) could be used to improve the quality of random samples of C-CCA.

Our non-parametric model in Chapter 5 could be extended to generate images in a coarse-to-fine fashion. Thus, the higher-resolution patches from the dataset could be used to both hallucinate high-frequency details and increase the spatial resolution of the synthesized images.

The pipeline in Chapter 5 passes the global constraint from the parametric model to non-parametric model in the form of an image in the "common" RGB colorspace. However, instead of images, C-CCA could be trained with a multi-channel feature representation of an image. This feature representation could be used for faster and more accurate patch lookups (*e.g.*, every patch could be represented with a code in the manifold of patches, similarly to semantic hashing [136]).

Evaluation Quantitative evaluation of the novelty and perceptual realism of the synthesized images remains an open problem. While we quantitatively evaluated non-parametric image synthesis of Chapter 3 and C-CCA in Chapter 4, quantitatively evaluating the realism of automatically synthesized images is difficult. As the desired metric is inherently psychophysical, such evaluation would require user studies, where a user could be queried to compare images before and after detail hallucination to confirm the benefit of the model. Moreover, the user could be asked to compare synthetic and real images and asked to choose preferable image. Reliably and reproducibly measuring the realism of the images through visual Turing tests is difficult due to the number of comparisons required for users to evaluate.

One approach of obtaining large quantities of comparisons is to approximate the human's responses to the visual Turing with a classifier. This is similar to Zhu *et al.* [251] who trained a discriminative classifier to differentiate between realistic and non-realistic *composite* images from a dataset of real and automatically composited images. However, our datasets have a very low number of images (200 – 450), so it is easy for the classifier to overfit the training data.

On the other hand, there are approaches, such as SSIM [11], that evaluate the quality of an image in comparison to a ground-truth reference image. However, they are not applicable for completely novel images, as the reference

images are not available. Although one can measure the quality of the generated image by conditioning the synthesis to be as similar as possible to some test set image and then use the test set image as reference, the performance, however, would not measure the quality of the model. Indeed, a significant error score implies that the images are dissimilar, but it does not necessarily imply that the model has failed to produce a photo-realistic image. The absence of a reference image is also a problem for scenarios when a user must generate an image similar to the image in their "mind's eye". In such scenarios performance of the model can only be measured by the author of the image. Moreover, even if the user is provided by a target image, the user's constraints may be incompatible with the target image, since drawing an object by looking at a photograph is not trivial. Hence, further research of quantitative measurement of perceptual realism is required.

Bibliography

[1] Umar Mohammed, Simon J. D. Prince, and Jan Kautz. Visio-lization: generating novel facial images. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 28, pages 57:1–57:8, New York, NY, USA, July 2009. ACM.

[2] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[3] Eero P. Simoncelli. Statistical models for images: Compression, restoration and synthesis. In *Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, volume 1, pages 673–678. IEEE, 1997.

[4] Eric Risser, Charles Han, Rozenn Dahyot, and Eitan Grinspun. Synthesizing structured image hybrids. *ACM Transactions on Graphics (TOG)*, 29(4):85, 2010.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[6] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-PIE. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–8. IEEE, 2008.

[7] Kieron Messer, Josef Kittler, Mohammad Sadeghi, Sebastien Marcel, Christine Marcel, Samy Bengio, Fabien Cardinaux, Conrad Sanderson, Jacek Czyz, Luc Vandendorpe, et al. Face verification competition on the XM2VTS database. In *Audio-and Video-Based Biometric Person Authentication*, pages 964–974. Springer, 2003.

[8] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active Appearance Models. In *Computer Vision (ECCV), IEEE European Conference on*, pages 484–498. Springer, 1998.

[9] Eagle Jones and Stefano Soatto. Layered Active Appearance Models. In *Computer Vision (ICCV), IEEE International Conference on*, volume 2, pages 1097–1102. IEEE, 2005.

[10] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[11] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[12] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. PMBP: PatchMatch belief propagation for correspondence field estimation. *International Journal of Computer Vision*, 110(1):2–13, 2014.

[13] Daniyar Turmukhambetov, Neill D.F. Campbell, Dan B Goldman, and Jan Kautz. Interactive sketch-driven image synthesis. *Computer Graphics Forum*, pages n/a–n/a, 2015.

[14] Daniyar Turmukhambetov, Neill D.F. Campbell, Simon J.D. Prince, and Jan Kautz. Modeling object appearance using Context-Conditioned Component Analysis. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, June 2015.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 586–591. IEEE, 1991.

[17] Timothy F. Cootes, Christopher J. Taylor, and Andreas Lanitis. Active Shape Models: Evaluation of a multi-resolution method for improving image search. In *Proceedings of the British Machine Vision Conference*, pages 32.1–32.10. BMVA Press, 1994. doi:10.5244/C.8.32.

[18] Timothy F. Cootes and Christopher J. Taylor. Active Shape Models - 'smart snakes'. In *Proceedings of the British Machine Vision Conference*, pages 266–275. Springer, 1992.

[19] Timothy F. Cootes, Christopher J. Taylor, David H. Cooper, and Jim Graham. Active Shape Models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[20] Timothy F. Cootes, Ed R. Baldock, and James Graham. An introduction to Active Shape Models. *Image processing and analysis*, pages 223–248, 2000.

[21] Oren Freifeld, Alexander Weiss, Silvia Zuffi, and Michael J. Black. Contour people: A parameterized model of 2D articulated human shape. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 639–646. IEEE, 2010.

[22] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24:3, pages 408–416. ACM, 2005.

[23] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[24] Sam T. Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[25] Simon J.D. Prince, Jonathan Warrell, James H. Elder, and Fatima M Felisberti. Tied factor analysis for face recognition across large pose differences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):970–984, 2008.

[26] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear

projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[27] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, 2000.

[28] Xiaogang Wang and Xiaoou Tang. Dual-space linear discriminant analysis for face recognition. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 564–569, 2004.

[29] Simon J.D. Prince and James H. Elder. Probabilistic linear discriminant analysis for inferences about identity. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1–8. IEEE, 2007.

[30] Dihong Gong, Zhifeng Li, Dahua Lin, Jianzhuang Liu, and Xiaoou Tang. Hidden factor analysis for age invariant face recognition. In *Computer Vision (ICCV), IEEE International Conference on*, pages 2872–2879. IEEE, 2013.

[31] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[32] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[33] Zoubin Ghahramani and Geoffrey E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

[34] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[35] Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[36] Neil Lawrence. Probabilistic non-linear Principal Component Analysis with Gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

[37] Laurens Van der Maaten and Geoffrey E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[38] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[39] Jacob Goldberger, Geoffrey E. Hinton, Sam T. Roweis, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 513–520. Curran Associates, Inc., 2004.

[40] Ruslan Salakhutdinov and Geoffrey E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics*, pages 412–419, 2007.

[41] Hongteng Xu and Hongyuan Zha. Manifold based face synthesis from sparse samples. In *Computer Vision (ICCV), IEEE International Conference on*, pages 2208–2215. IEEE, 2013.

[42] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using Laplacianfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):328–340, 2005.

[43] Gareth J. Edwards, Christopher J. Taylor, and Timothy F. Cootes. Interpreting face images using Active Appearance Models. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 300–305. IEEE, 1998.

[44] Iain Matthews and Simon Baker. Active Appearance Models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004.

[45] Der-Tsai Lee and Bruce J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[46] Ian M. Scott, Timothy F. Cootes, and Christopher J. Taylor. Improving Appearance Model matching using local image structure. In *Information Processing in Medical Imaging*, pages 258–269. Springer, 2003.

[47] Timothy F. Cootes and Christopher J. Taylor. An algorithm for tuning an Active Appearance Model to new data. In *Proceedings of the British Machine Vision Conference*, pages 919–928, 2006.

[48] Patrick Sauer, Timothy F. Cootes, and Christopher J. Taylor. Accurate regression procedures for Active Appearance Models. In *Proceedings of the British Machine Vision Conference*, pages 1–11, 2011.

[49] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[50] Ankur Patel and William A.P. Smith. 3D morphable face models revisited. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1327–1334. IEEE, 2009.

[51] Tanasai Sucontphunt, Borom Tunwattanapong, Zhigang Deng, and Ulrich Neumann. Crafting 3D faces using free form portrait sketching and plausible texture inference. In *Proceedings of Graphics Interface 2010*, pages 209–216. Canadian Information Processing Society, 2010.

[52] Tanasai Sucontphunt and Ulrich Neumann. 3D facial surface and texture synthesis using 2D landmarks from a single face sketch. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, pages 152–159, Oct 2012.

[53] Kevin N. Walker, Timothy F. Cootes, and Christopher J. Taylor. Automatically building Appearance Models from image sequences using salient features. *Image and Vision Computing*, 20(5):435–440, 2002.

[54] Krishnan Ramnath, Simon Baker, Iain Matthews, and Deva Ramanan. Increasing the density of Active Appearance Models. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1–8. IEEE, 2008.

[55] Julia Krüger, Jan Ehrhardt, and Heinz Handels. Probabilistic appearance models for segmentation and classification. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1698–1706, 2015.

[56] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics (TOG)*, volume 24:3, pages 1134–1141. ACM, 2005.

[57] Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. Character animation from 2D pictures and 3D motion data. *ACM Transactions on Graphics (TOG)*, 26(1):1, 2007.

[58] Pei Zhang and Timothy F. Cootes. Automatic construction of parts+ geometry models for initializing groupwise registration. *Medical Imaging, IEEE Transactions on*, 31(2):341–358, 2012.

[59] Steve A. Adeshina and Timothy F. Cootes. Constructing part-based models for groupwise registration. In *Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on*, pages 1073–1076. IEEE, 2010.

[60] Erik G. Learned-Miller. Data driven image models through continuous joint alignment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):236–250, 2006.

[61] Timothy F. Cootes, Carole J. Twining, Vladimir S. Petrovic, Roy Schestowitz, and Christopher J. Taylor. Groupwise construction of appearance models using piece-wise affine deformations. In *Proceedings of the British Machine Vision Conference*, volume 5, pages 879–888, 2005.

[62] Steve A. Adeshina and Timothy F. Cootes. Evaluation of performance of part-based models for groupwise registration. In *Medical Image Understanding and Analysis*, 2010.

[63] René Donner, Horst Wildenauer, Horst Bischof, and Georg Langs. Weakly supervised group-wise model learning based on discrete optimization. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009*, pages 860–868. Springer, 2009.

[64] Serdar K. Balci, Polina Golland, Martha Shenton, and William M. Wells. Free-form B-spline deformation model for groupwise registration. In *Medical image computing and computer-assisted intervention: MICCAI... International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 10, page 23. NIH Public Access, 2007.

[65] Timothy F. Cootes, Carole J. Twining, Vladimir S. Petrović, Kolawole O. Babalola, and Christopher J. Taylor. Computing accurate correspondences across groups of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(11):1994–2005, 2010.

[66] Min-Jeong Kim, Myoung-Hee Kim, and Dinggang Shen. Learning-based deformation estimation for fast non-rigid registration. In *Computer Vision and Pattern Recognition Workshops (CVPRW), The IEEE International Conference on*, pages 1–6. IEEE, 2008.

[67] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. SIFT flow: Dense correspondence across different scenes. In *Computer Vision (ECCV), IEEE European Conference on*, pages 28–42. Springer, 2008.

[68] David G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[69] Brendan J. Frey and Nebojsa Jojic. Transformed Component Analysis: joint estimation of spatial transformations and image components. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1190–1196, 1999.

[70] John Winn and Nebojsa Jojic. LOCUS: Learning object classes with unsupervised segmentation. In *Computer Vision, International Conference on*, volume 1, pages 756–763. IEEE, 2005.

[71] Hossein Mobahi, Ce Liu, and William T. Freeman. A compositional model for low-dimensional image set representation. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*. IEEE Computer Society, 2014.

[72] Thomas J. Cashman and Andrew Fitzgibbon. What shape are dolphins? building 3D morphable models from 2D images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):232–244, 2013.

[73] Sara Vicente, João Carreira, Lourdes Agapito, and Jorge Batista. Reconstructing Pascal VOC. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 41–48. IEEE, 2014.

[74] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://host.robots.ox.ac.uk/pascal/VOC/voc2010/index.html. Accessed: 2015-07-17.

[75] Manuel Marques and João Costeira. Estimating 3d shape from degenerate sequences with missing data. *Computer Vision and Image Understanding*, 113(2):261–272, 2009.

[76] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994.

[77] Abhishek Kar, Shubham Tulsiani, João Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1966–1974. IEEE, 2015.

[78] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):878–892, 2008.

[79] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30, 2012.

[80] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[81] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[82] Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, 100(1):67–92, 1973.

[83] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[84] M. Pawan Kumar, Philip H.S. Torr, and Andrew Zisserman. Extending pictorial structures for object recognition. In *Proceedings of the British Machine Vision Conference*, pages 81–1, 2004.

[85] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, volume 2, pages 66–73. IEEE, 2000.

[86] M. Pawan Kumar, Philip H.S. Torr, and Andrew Zisserman. Learning layered pictorial structures from video. In *ICVGIP*, pages 158–164, 2004.

[87] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, volume 1, pages 886–893. IEEE, 2005.

[88] Hamed Pirsiavash and Deva Ramanan. Steerable part models. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 3226–3233. IEEE, 2012.

[89] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1385–1392. IEEE, 2011.

[90] Silvia Zuffi, Oren Freifeld, and Michael J. Black. From pictorial structures to deformable structures. In *Computer Vision and Pattern*

Recognition (CVPR), The IEEE International Conference on, pages 3546–3553. IEEE, 2012.

[91] Michael C. Burl, Thomas K. Leung, and Pietro Perona. Face localization via shape statistics. In *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, pages 154–159, 1995.

[92] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision (ECCVW), IEEE European Conference on*, pages 17–32, 2004.

[93] Markus Weber, Wolfgang Einhäuser, Max Welling, and Pietro Perona. Viewpoint-invariant learning and detection of human heads. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 20–27. IEEE, 2000.

[94] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, volume 2, pages II–264. IEEE, 2003.

[95] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *Computer Vision (ECCV), IEEE European Conference on*, pages 18–32, 2000.

[96] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 2879–2886. IEEE, 2012.

[97] Ying Nian Wu, Zhangzhang Si, Chuck Fleming, and Song-Chun Zhu. Deformable template as active basis. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1–8. IEEE, 2007.

[98] Ying Nian Wu, Zhangzhang Si, Haifeng Gong, and Song-Chun Zhu. Learning active basis model for object detection and recognition. *International journal of computer vision*, 90(2):198–235, 2010.

[99] Zhangzhang Si and Ying Nian Wu. Wavelet, active basis, and shape script: A tour in the sparse land. In *Proceedings of the International*

Conference on Multimedia Information Retrieval, MIR '10, pages 201–210, New York, NY, USA, 2010. ACM.

[100] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1365–1372. IEEE, 2009.

[101] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[102] Yang Wang, Duan Tran, and Zicheng Liao. Learning hierarchical poselets for human parsing. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1705–1712. IEEE, 2011.

[103] Leonid Pishchulin, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Poselet conditioned pictorial structures. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 588–595. IEEE, 2013.

[104] Silvio Savarese and Li Fei-Fei. View synthesis for recognizing unseen poses of object classes. In *Computer Vision (ECCV), IEEE European Conference on*, pages 602–615. Springer, 2008.

[105] Silvio Savarese and Fei-Fei Li. 3D generic object categorization, localization and pose estimation. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1–8, 2007.

[106] Min Sun, Hao Su, Silvio Savarese, and Li Fei-Fei. A multi-view probabilistic model for 3D object classes. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1247–1254, June 2009.

[107] Silvio Savarese and Li Fei-Fei. Multi-view object categorization and pose estimation. In *Computer Vision*, pages 205–231. Springer, 2010.

[108] Hao Su, Min Sun, Li Fei-Fei, and Silvio Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 213–220. IEEE, 2009.

[109] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision (ECCV), IEEE European Conference on*, pages 73–86. Springer, 2012.

[110] Tinghui Zhou, Yong Jae Lee, Stella X Yu, and Alexei A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 1191–1200, 2015.

[111] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Context as supervisory signal: Discovering objects with predictable context. In *Computer Vision (ECCV), IEEE European Conference on*, pages 362–377. Springer, 2014.

[112] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *Advances in Neural Information Processing Systems (NIPS)*, pages 494–502, 2013.

[113] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes Paris look like Paris? *ACM Transactions on Graphics (TOG)*, 31(4):101:1–101:9, 2012.

[114] Yong Jae Lee, Alexei A. Efros, Martial Hebert, et al. Style-aware mid-level representation for discovering visual connections in space and time. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1857–1864. IEEE, 2013.

[115] Stefan Lee, Nicolas Maisonneuve, David Crandall, Alexei A. Efros, and Josef Sivic. Linking past to present: Discovering style in two centuries of architecture. In *IEEE International Conference on Computational Photography (ICCP)*, 2015.

[116] Jun-Yan Zhu, Yong Jae Lee, and Alexei A. Efros. AverageExplorer: Interactive exploration and alignment of visual data collections. *ACM Transactions on Graphics (TOG)*, 33(4), 2014.

[117] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55, 2012.

[118] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Transactions on Graphics (TOG)*, 30(4):33:1–33:10, July 2011.

[119] Chuan Li, Oliver Deussen, Yi-Zhe Song, Phil Willis, and Peter Hall. Modeling and generating moving trees from video. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, pages 127:1–127:12, New York, NY, USA, 2011. ACM.

[120] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth Teller. Reconstructing 3D tree models from instrumented photographs. *Computer Graphics and Applications, IEEE*, 21(3):53–61, May 2001.

[121] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, New York, NY, USA, 2007. ACM.

[122] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, New York, NY, USA, 2007. ACM.

[123] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 599–604. ACM, 2006.

[124] Bela Julesz. Visual pattern discrimination. *Information Theory, IRE Transactions on*, 8(2):84–92, 1962.

[125] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[126] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[127] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[128] Leon Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 262–270. Curran Associates, Inc., 2015.

[129] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[130] Chuan Li and Michael Wand. Combining Markov random fields and convolutional neural networks for image synthesis. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, June 2016.

[131] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[132] Dosovitskiy Alexey, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2015.

[133] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2015.

[134] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), IEEE International Conference on*, pages 2018–2025. IEEE, 2011.

[135] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan Russell, and Josef Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2014.

[136] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[137] Yoav Freund and David Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

[138] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

[139] S.M. Ali Eslami, Nicolas Heess, and John Winn. The shape Boltzmann machine: a strong model of object shape. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 406–413. IEEE, 2012.

[140] Ruslan R. Salakhutdinov and Geoffrey E. Hinton. Deep Boltzmann machines. In *Proceedings of the international conference on artificial intelligence and statistics*, volume 5:2, pages 448–455. MIT Press Cambridge, MA, 2009.

[141] S.M. Ali Eslami and Christopher K.I. Williams. A generative model for parts-based object segmentation. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 100–107. Curran Associates, Inc., 2012.

[142] S.M.Ali Eslami, Nicolas Heess, Christopher K.I. Williams, and John Winn. The shape boltzmann machine: A strong model of object shape. *International Journal of Computer Vision*, 107:155–176, 2014.

[143] Chi Nhan Duong, Khoa Luu, Kha Gia Quach, and Tien D. Bui. Beyond Principal Components: Deep Boltzmann machines for face modeling. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, June 2015.

[144] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[145] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[146] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

[147] Alex Krizhevsky and Geoffrey E. Hinton. Learning multiple layers of features from tiny images, 2009.

[148] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[149] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[150] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision (ICCV), IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[151] Claude Elwood Shannon and Warren Weaver. A mathematical theory of communication, 1948.

[152] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[153] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM, 2001.

[154] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3):24, 2009.

[155] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[156] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 22:3, pages 277–286. ACM, 2003.

[157] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[158] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized PatchMatch correspondence algorithm. In *Computer Vision (ECCV), IEEE European Conference on*, pages 29–43. Springer, 2010.

[159] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):82, 2012.

[160] Connelly Barnes, Fang-Lue Zhang, Liming Lou, Xian Wu, and Shi-Min Hu. PatchTable: Efficient patch queries for large datasets and applications. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Aug 2015.

[161] Shi-Min Hu, Fang-Lue Zhang, Miao Wang, Ralph R. Martin, and Jue Wang. PatchNet: A patch-based image representation for interactive library-driven image editing. *ACM Transactions on Graphics (TOG)*, 32:196:1–196:12, nov 2013.

[162] Lubin Fan, Przemyslaw Musialski, Ligang Liu, and Peter Wonka. Structure completion for facade layouts. *ACM Transactions on Graphics (TOG)*, 33(6):210:1–210:11, November 2014.

[163] Dengxin Dai, Hayko Riemenschneider, Gilbert Schmitt, and Luc Van. Example-based facade texture synthesis. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1065–1072. IEEE, 2013.

[164] Sylvain Lefebvre, Samuel Hornus, and Anass Lasram. By-example synthesis of architectural textures. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 84:1–84:8, New York, NY, USA, 2010. ACM.

[165] Sawsan AlHalawani, Yong-Liang Yang, Han Liu, and Niloy J Mitra. Interactive facades analysis and synthesis of semi-regular facades. In *Computer Graphics Forum*, volume 32, pages 215–224. Wiley Online Library, 2013.

[166] Duygu Ceylan, Niloy J. Mitra, Youyi Zheng, and Mark Pauly. Coupled structure-from-motion and 3D symmetry detection for urban facades. *ACM Transactions on Graphics (TOG)*, 33(1):2, 2014.

[167] Minh Dang, Duygu Ceylan, Boris Neubert, and Mark Pauly. SAFE: Structure-aware facade editing. *Computer Graphics Forum*, 33(2):83–93, 2014.

[168] Lexing Ying, Aaron Hertzmann, Henning Biermann, and Denis Zorin. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Conference on Rendering*, EGWR'01, pages 301–312, Aire-la-Ville, Switzerland, Switzerland, 2001. Eurographics Association.

[169] Hui Fang and John C. Hart. Textureshop: Texture synthesis as a photograph editing tool. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 354–359, New York, NY, USA, 2004. ACM.

[170] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 364–367, New York, NY, USA, 2004. ACM.

[171] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 327–340, 2001.

[172] Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *ACM Transactions on Graphics (TOG)*, 34(2), 2015.

[173] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. In *ACM Transactions on Graphics (TOG)*, volume 24:3, pages 777–786. ACM, 2005.

[174] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *ACM Transactions on Graphics (TOG)*, volume 25:3, pages 541–548. ACM, 2006.

[175] Marshall F. Tappen and Ce Liu. A Bayesian approach to alignment-based image hallucination. In *Computer Vision (ECCV), IEEE European Conference on*, pages 236–249. Springer, 2012.

[176] Nebojsa Jojic, Brendan J. Frey, and Anitha Kannan. Epitomic analysis of appearance and shape. In *Computer Vision (ICCV), IEEE International Conference on*, pages 34–41. IEEE, 2003.

[177] Nebojsa Jojic and Yaron Caspi. Capturing image structure with probabilistic index maps. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, volume 1, pages I–212. IEEE, 2004.

[178] Anitha Kannan, John Winn, and Carsten Rother. Clustering appearance and shape by learning jigsaws. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, page 657. MIT; 1998, 2007.

[179] Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew Fitzgibbon. Unwrap mosaics: a new representation for video editing. In *ACM Transactions on Graphics (TOG)*, volume 27:3, page 17. ACM, 2008.

[180] James Hays and Alexei A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26:3, page 4. ACM, 2007.

[181] Micah K. Johnson, Kevin Dale, Shai Avidan, Hanspeter Pfister, William T. Freeman, and Wojciech Matusik. CG2Real: Improving the realism of computer generated images using a large collection of photographs. *Visualization and Computer Graphics, IEEE Transactions on*, 17(9):1273–1285, 2011.

[182] Chen Goldberg, Tao Chen, Fang-Lue Zhang, Ariel Shamir, and Shi-Min Hu. Data-driven object manipulation in images. In *Computer Graphics Forum*, volume 31:2:1, pages 265–274. Wiley Online Library, 2012.

[183] Jean-François Lalonde, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. In *ACM Transactions on Graphics (TOG)*, volume 26:3, page 3. ACM, 2007.

[184] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models. *ACM Transactions on Graphics (TOG)*, 32(4):123:1–123:12, 2013.

[185] Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T. Freeman. Matching and predicting street level images. In *Workshop on Vision for Cognitive Tasks in Computer Vision (ECCVW), IEEE European Conference on*, 2010.

[186] Phillip Isola and Ce Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *Computer Vision (ICCV), IEEE International Conference on*, pages 3048–3055. IEEE, 2013.

[187] Matthew Johnson, Gabriel J. Brostow, Jamie Shotton, Ognjen Arandjelovic, Vivek Kwatra, and Roberto Cipolla. Semantic photo synthesis. *Computer Graphics Forum*, 25(3):407–413, 2006.

[188] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2Photo: internet image montage. In *ACM Transactions on Graphics (TOG)*, volume 28:5, page 124. ACM, 2009.

[189] Tao Chen, Ping Tan, Li-Qian Ma, Ming-Ming Cheng, Ariel Shamir, and Shi-Min Hu. PoseShop: Human image database construction and personalized content synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):824–837, May 2013.

[190] Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketcher: interactive sketch-based image synthesis. *Computer Graphics and Applications, IEEE*, 31(6):56–66, 2011.

[191] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *Visualization and Computer Graphics, IEEE Transactions on*, 17(11):1624–1636, 2011.

[192] Yen-Liang Lin, Cheng-Yu Huang, Hao-Jeng Wang, and Wei-Chou Hsu. 3D sub-query expansion for improving sketch-based multi-view image retrieval. In *Computer Vision (ICCV), IEEE International Conference on*, pages 3495–3502. IEEE, 2013.

[193] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011.

[194] Ming-Ming Cheng, Guo-Xin Zhang, Niloy J. Mitra, Xiaolei Huang, and Shi-Min Hu. Global contrast based salient region detection. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, pages 409–416. IEEE, 2011.

[195] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[196] Natasha Kholgade, Tomas Simon, Alexei A. Efros, and Yaser Sheikh. 3D object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4), 2014.

[197] Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. ShadowDraw: real-time user guidance for freehand drawing. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 27:1–27:10, New York, NY, USA, 2011. ACM.

[198] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics (TOG)*, 23(3):294–302, 2004.

[199] Ira Kemelmacher-Shlizerman, Aditya Sankar, Eli Shechtman, and Steven M. Seitz. Being John Malkovich. In *Computer Vision (ECCV), IEEE European Conference on*, pages 341–353. Springer, 2010.

[200] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M. Seitz. Exploring photobios. In *ACM Transactions on Graphics (TOG)*, volume 30, page 61. ACM, 2011.

[201] Kevin Dale, Kalyan Sunkavalli, Micah K. Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. Video face replacement. *ACM Transactions on Graphics (TOG)*, 30(6):130, 2011.

[202] Fei Yang, Jue Wang, Eli Shechtman, Lubomir Bourdev, and Dimitri Metaxas. Expression flow for 3D-aware face component transfer. *ACM Transactions on Graphics (TOG)*, 30(4):60, 2011.

[203] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003.

[204] Ce Liu, Heung-Yeung Shum, and Chang-Shui Zhang. A two-step approach to hallucinating faces: global parametric model and local nonparametric model. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, volume 1, pages I–192. IEEE, 2001.

[205] Ce Liu, Heung-Yeung Shum, and William T. Freeman. Face hallucination: Theory and practice. *International Journal of Computer Vision*, 75(1):115–134, 2007.

[206] Pouria Mortazavian, Josef Kittler, William J. Christmas, and Uk Guildford. 3D-assisted facial texture super-resolution. In *Proceedings of the British Machine Vision Conference*, pages 1–11, 2009.

[207] Simon Baker and Tekeo Kanade. Limits on super-resolution and how to break them. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(9):1167–1183, 2002.

[208] Arnaud Dessein, William A.P. Smith, Richard C. Wilson, and Edwin R. Hancock. Example-based modeling of facial texture from deficient data. In *Computer Vision (ICCV), IEEE International Conference on*, pages 3898–3906, 2015.

[209] Daniel Dixon, Manoj Prasad, and Tracy Hammond. iCanDraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 897–906, 2010.

[210] Yotam Gingold, Takeo Igarashi, and Denis Zorin. Structured annotations for 2d-to-3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, page 148. ACM, 2009.

[211] William H. McMaster. Polarization and the Stokes parameters. *American Journal of Physics*, 22:351–362, 1954.

[212] Frank P Kuhl and Charles R. Giardina. Elliptic Fourier features of a closed contour. *Computer Graphics and Image Processing*, 18:236–258, 1982.

[213] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. Sketch-sketch revolution: An engaging tutorial system for guided sketching and application learning. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 373–382. ACM, 2011.

[214] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. The drawing assistant: Automated drawing guidance and feedback from photographs. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, pages 183–192, 2013.

[215] Preston Blair. *Cartoon animation (the collector's series)*. Collector's Series. Walter Foster Publishing, 1994.

[216] Koos Eissen and Roselien Steur. *Sketching: Drawing techniques for product designers*. BIS Publishers, 2007.

[217] Grant Fuller. *Start sketching and drawing now*. North Light Books, 2011.

[218] DrawingNow website.

`http://www.drawingnow.com/how-to-draw-animals.html`.

[219] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *Computer Vision (ECCV), IEEE European Conference on*, pages 109–122. Springer, 2002.

[220] Eran Borenstein and Shimon Ullman. Learning to segment. In *Computer Vision (ECCV), IEEE European Conference on*, pages 315–328. Springer, 2004.

[221] Eran Borenstein, Eitan Sharon, and Shimon Ullman. Combining top-down and bottom-up segmentation. In *In Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR*, volume 4, page 46, 2004.

[222] Andrew Fitzgibbon, Maurizio Pilu, and Robert B Fisher. Direct least square fitting of ellipses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):476–480, 1999.

[223] Victor Adrian Prisacariu and Ian Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2011.

[224] Ramanan Navaratnam, Andrew Fitzgibbon, and Roberto Cipolla. The joint manifold model for semi-supervised multi-valued regression. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1–8. IEEE, 2007.

[225] Harry G. Barrow, Jay M. Tenenbaum, Robert C. Bolles, and Helen C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. of the International Joint Conference of Artificial Intelligence*, pages 659–663, 1977.

[226] John Brooke. SUS-a quick and dirty usability scale. *Usability evaluation in industry*, 189:194, 1996.

[227] James R. Lewis and Jeff Sauro. The factor structure of the system usability scale. In *Human Centered Design*, pages 94–103. Springer, 2009.

[228] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *CoRR*, abs/1406.2031, 2014.

[229] Andrea Vedaldi, Siddarth Mahendran, Stavros Tsogkas, Subhransu Maji, Ross B. Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B. Blaschko, David Weiss, Ben Taskar, Karen Simonyan, Naomi Saphra, and Sammy Mohamed. Understanding objects in detail with fine-grained attributes. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2014.

[230] Stephen Gould and Yuhang Zhang. PatchMatchGraph: Building a graph of dense patch correspondences for label transfer. In *Computer Vision (ECCV), IEEE European Conference on*, pages 439–452. Springer, 2012.

[231] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. DNN flow: DNN feature pyramid based image matching. In *Proceedings of the British Machine Vision Conference*, 2014.

[232] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), The IEEE International Conference on*, 2011.

[233] Timothy F. Cootes, Carole J. Twining, Vladimir S. Petrovic, Kolawole O. Babalola, and Christopher J. Taylor. Computing accurate correspondences across groups of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(11):1994–2005, Nov 2010.

[234] Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference.* Cambridge University Press, 2012.

[235] Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[236] Evelyn Fix and Joseph L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

[237] Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244, 2001.

[238] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[239] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[240] Olivier Teboul. Ecole Centrale Paris Facades Database. http://vision.mas.ecp.fr/Personnel/teboul/data.php. Accessed: 2014-07-01.

[241] Boris Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[242] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

[243] M. Alex O. Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision (ECCV), IEEE European Conference on*, pages 447–460. Springer, 2002.

[244] M. Alex O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 456–460. IEEE, 2002.

[245] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th international conference on Machine learning*, pages 975–982. ACM, 2007.

[246] Ken Shoemake. Animating rotation with quaternion curves. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 19, pages 245–254. ACM, 1985.

[247] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[248] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

[249] Alexander T. Ihler and David A. McAllester. Particle belief propagation. In *International Conference on Artificial Intelligence and Statistics*, pages 256–263, 2009.

[250] Rajkumar Kothapa, Jason Pacheco, and Erik B. Sudderth. Max-product particle belief propagation. *Master's project report, Brown University Dept. of Computer Science*, 2011.

[251] Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A. Efros. Learning a Discriminative model for the Perception of Realism in Composite Images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3943–3951, 2015.

Appendix A

Dataset Examples

We have compiled four datasets:

- We have labeled 328 images of Weizmann horse database [219, 220, 221] that already has segmentation masks. The labels are: head, neck, torso, 4 legs, tail. We removed duplicates and images with severe visual artifacts to a reduced set of 295 images. See Figure A.1.
- We downloaded 281 images of Elephants from internet, manually segmented them, and labeled with following labels: head, torso, 4 legs, trunk. See Figure A.2.
- We downloaded 270 images of Pigeons from internet, manually segmented them, and labeled with following labels: head, neck, torso, wing, 2 legs, tail. See Figure A.3.
- We have purchased a photomontage of cats captured on white background by professional photographer. The photomontage has 392 images of Cats. The license does not allow sharing of the photomontage, so we can only publicly show derived results. The images were segmented from the white background and manually labeled. The labels are: head, torso, tail and 4 legs. See Figure A.4.

All images were flipped, such that the animal is looking in the front to left direction. Examples are shown in the Appendix A.

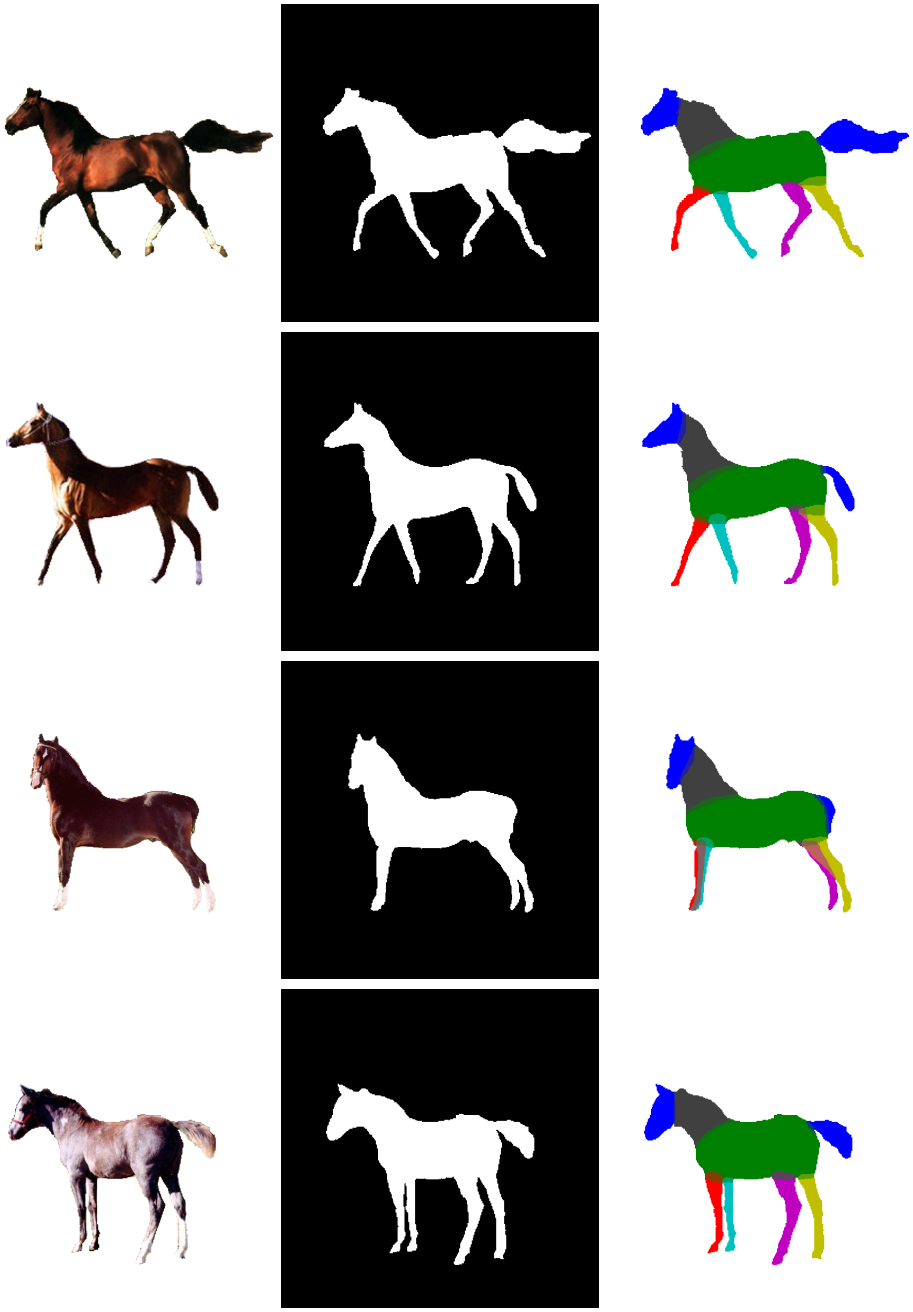


Figure A.1: Examples from Horses dataset. From left to right: Image, Segmentation, Part Labeling.

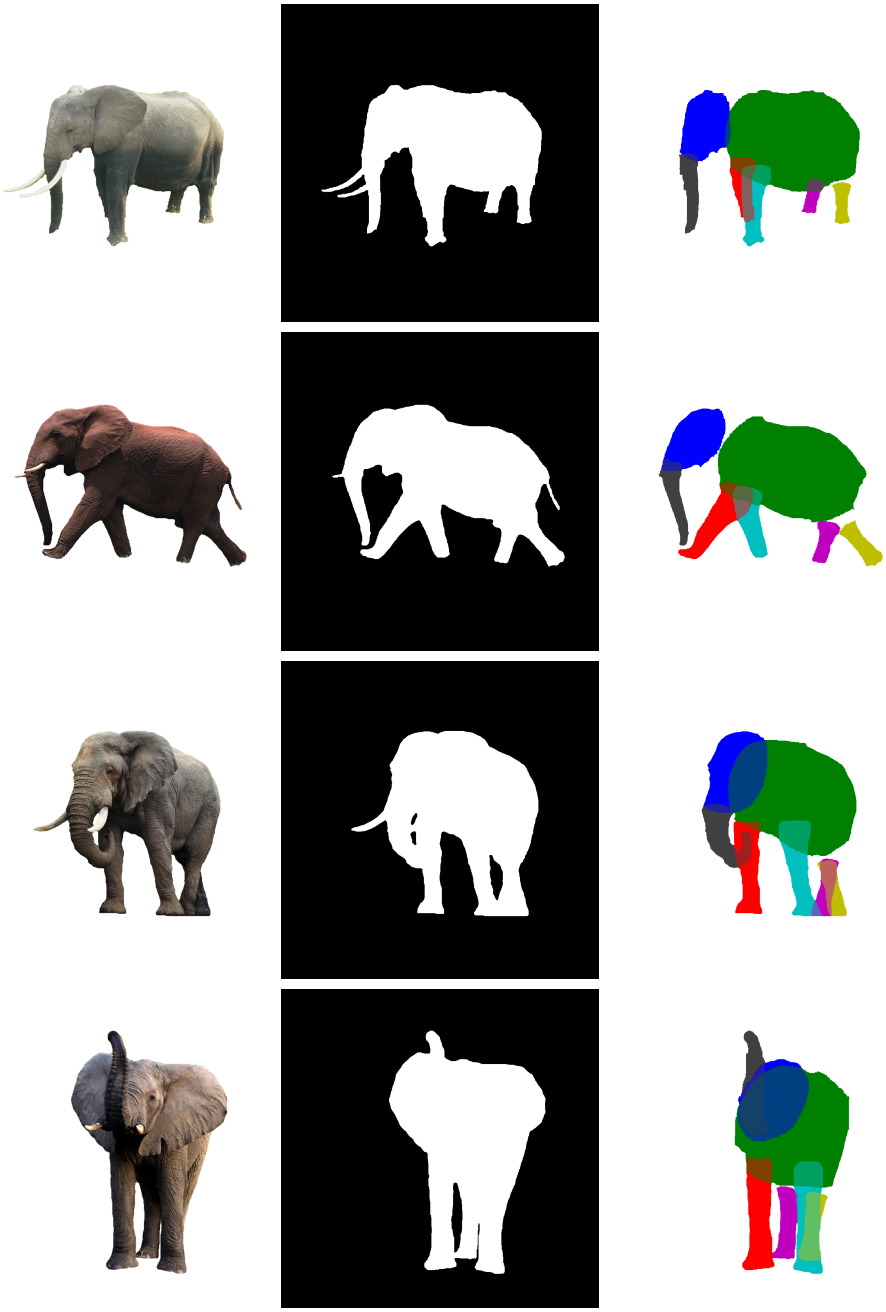


Figure A.2: Examples from Elephants dataset. From left to right: Image, Segmentation, Part Labeling.

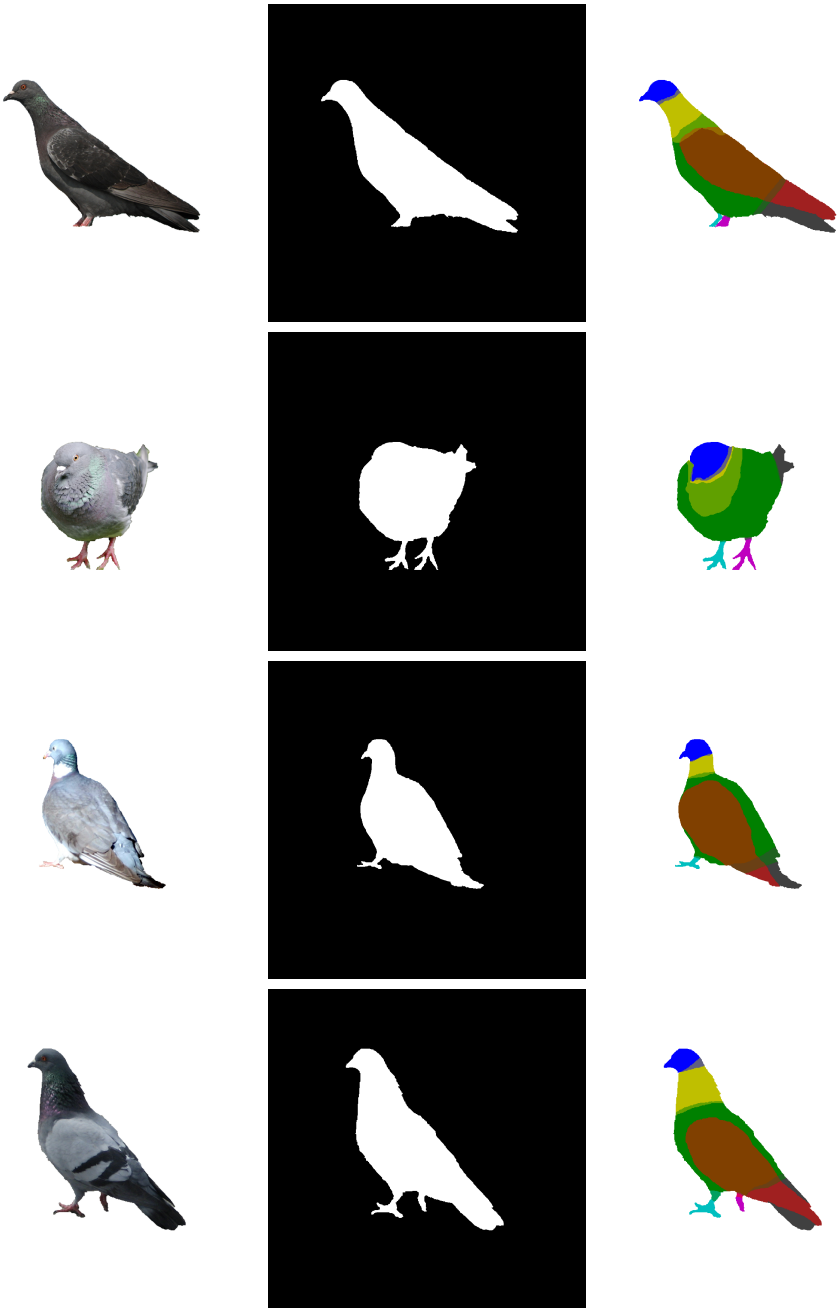


Figure A.3: Examples from Pigeons dataset. From left to right: Image, Segmentation, Part Labeling.

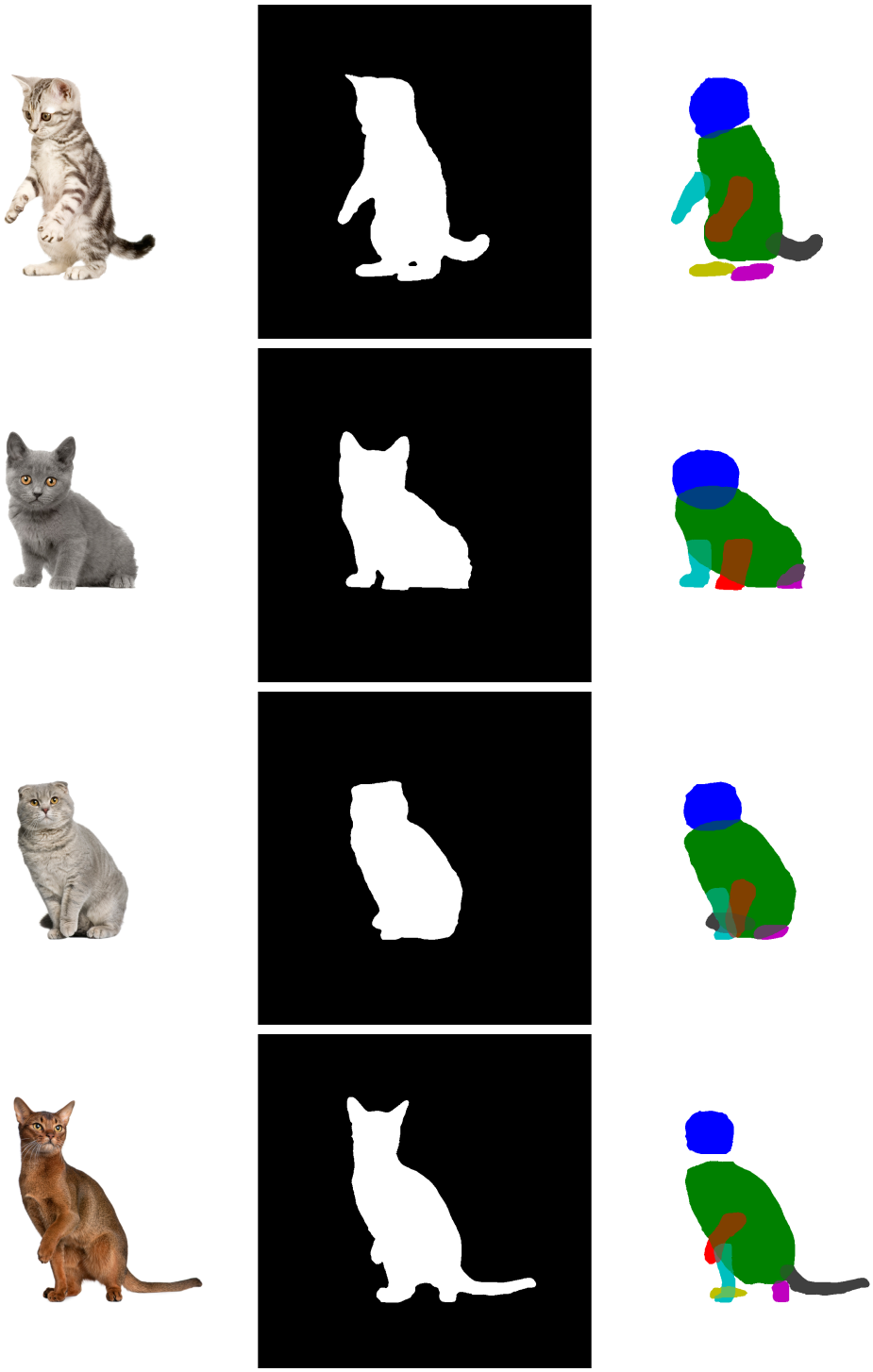


Figure A.4: Examples from Cats dataset. From left to right: Image, Segmentation, Part Labeling.

Appendix B

Interactive Sketch-Driven Image Synthesis System User Interface

Figures B.1 to B.7 show different tools of the Interactive Sketch-Driven Image Synthesis system interface described in Chapter 3.

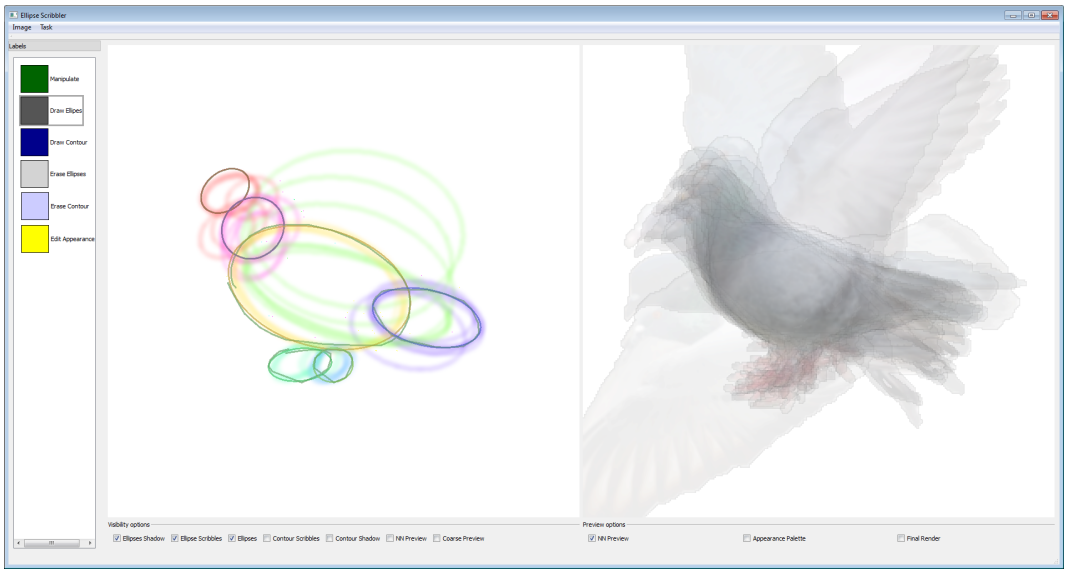


Figure B.1: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Draw Ellipses". The user has scribbled a few masses. The panel on the left shows system's suggestions for masses, *i.e.* possible places to draw next ellipse. The panel on the right shows an averaged image of pigeons that have similar mass configuration.

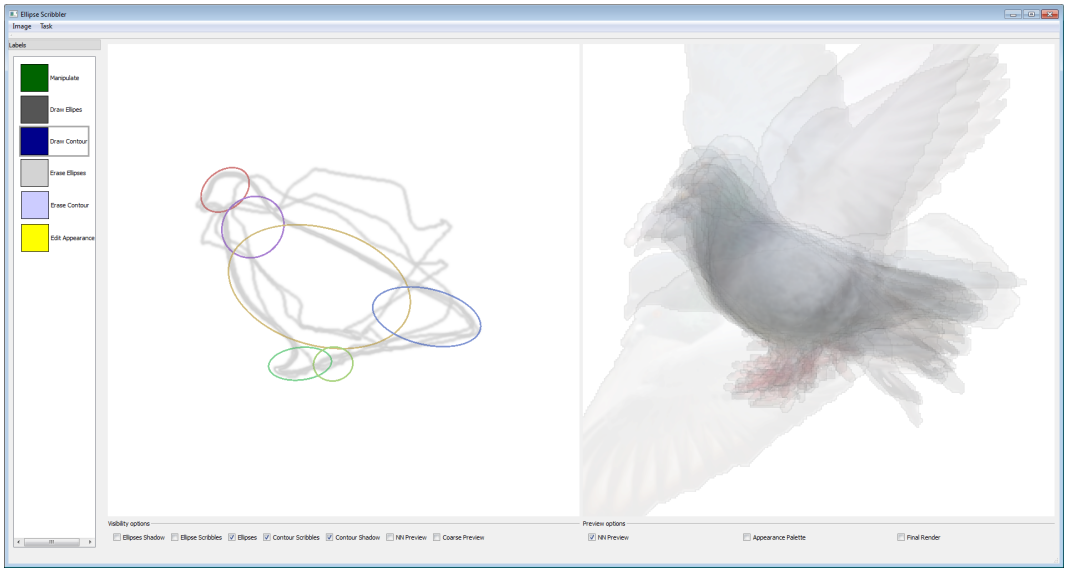


Figure B.2: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Draw Contour". The user has specified some ellipses but hasn't started drawing the contour. The panel on the left shows system's suggestions for the contour, *i.e.* possible contours of the pigeon given the previous ellipse configuration. The panel on the right shows an averaged image of pigeons that have similar mass configuration.

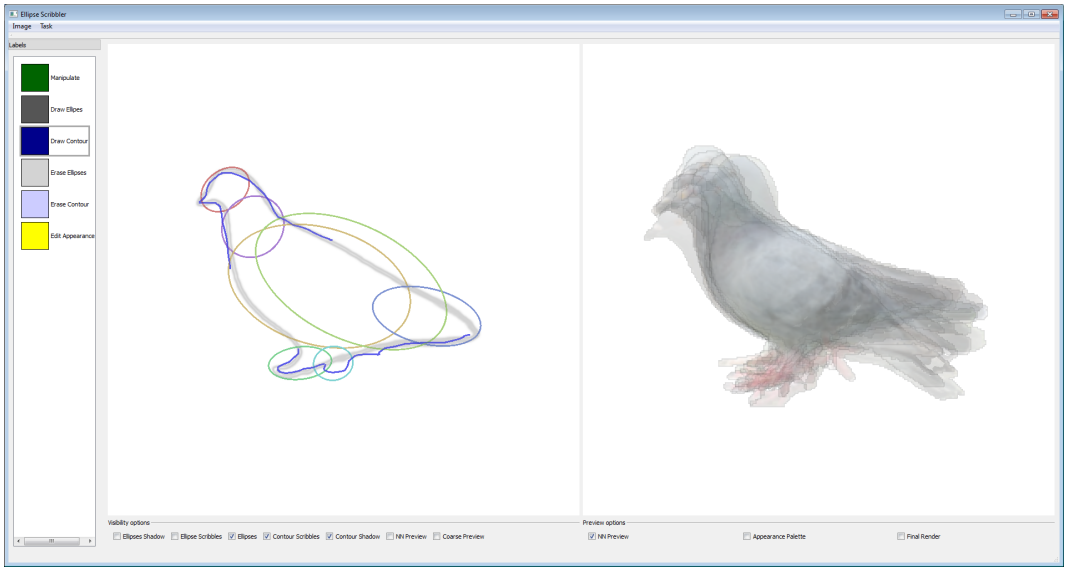


Figure B.3: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Draw Contour". The user has specified some ellipses and drawn parts of the contour. The panel on the left shows system's suggestions for the contour, *i.e.* possible contours of the pigeon given the previous ellipse configuration and the sketched contour. The panel on the right shows an averaged image of pigeons that have similar mass configuration and partial contour.

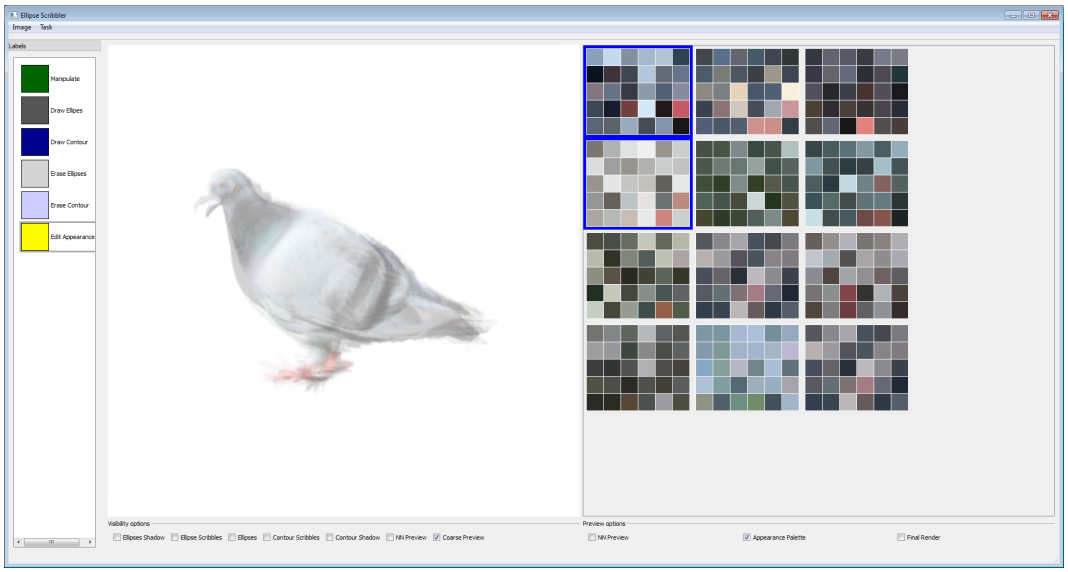


Figure B.4: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Edit Appearance". The user has specified some ellipses and has drawn parts of the contour. The panel on the left shows preview of the system's output given the mass configuration, partial contour and selected appearance palettes on the right. The panel on the right shows appearance palettes of pigeons that have similar mass configuration and contour.

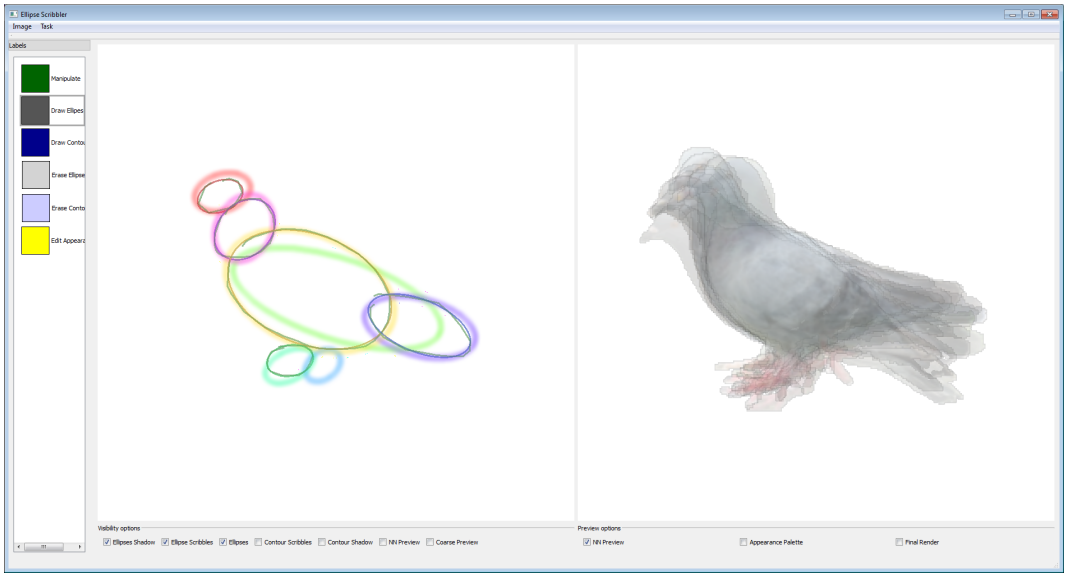


Figure B.5: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Draw Ellipse". The user has previously specified some ellipses and has drawn parts of the contour and now wants the refine the specifications by scribbling more ellipses. The panels show similar content to Figure B.1.

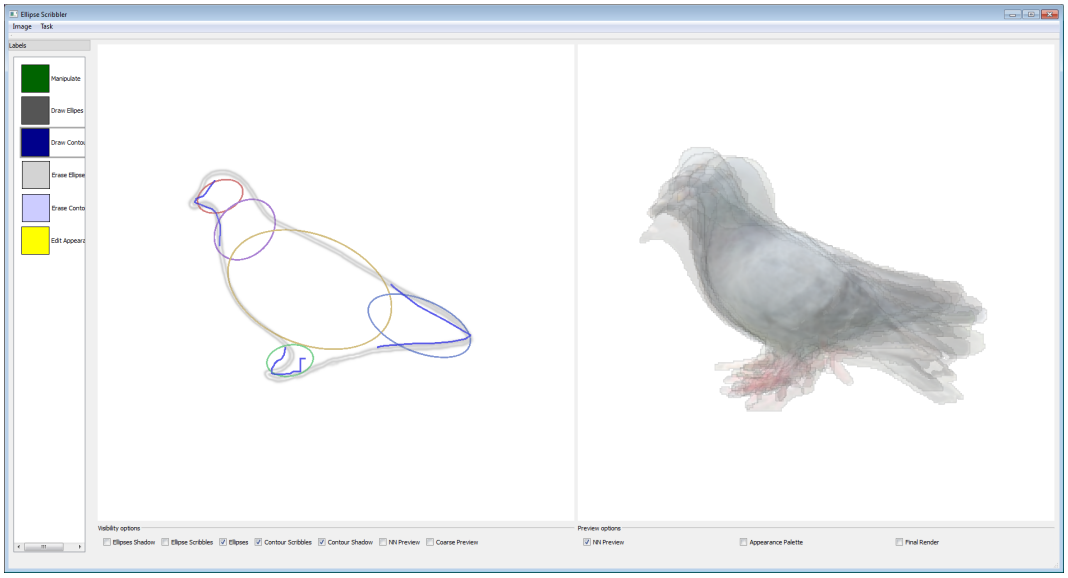


Figure B.6: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Draw Contour". The user has previously specified some ellipses and has drawn parts of the contour and now wants the refine the contour. The panels show similar content to Figure B.3.

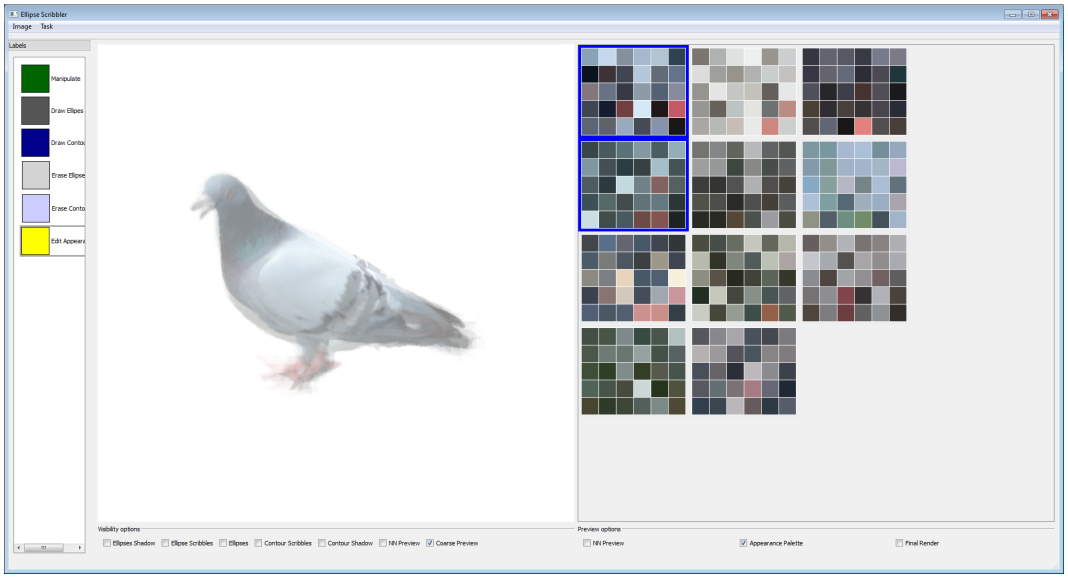


Figure B.7: Interactive Sketch-Driven Image Synthesis System User Interface. The active tool is "Edit Appearance". The user has previously specified some ellipses and has drawn parts of the contour. The panels show similar content to Figure B.4. Since the specifications of the user have changed, the selection of appearance palettes is now different.

Appendix C

User Sketch Examples

Figures C.1 to C.3 show examples of the sketches and synthesized results created using the Interactive Sketch-Driven Image Synthesis system described in Chapter 3.

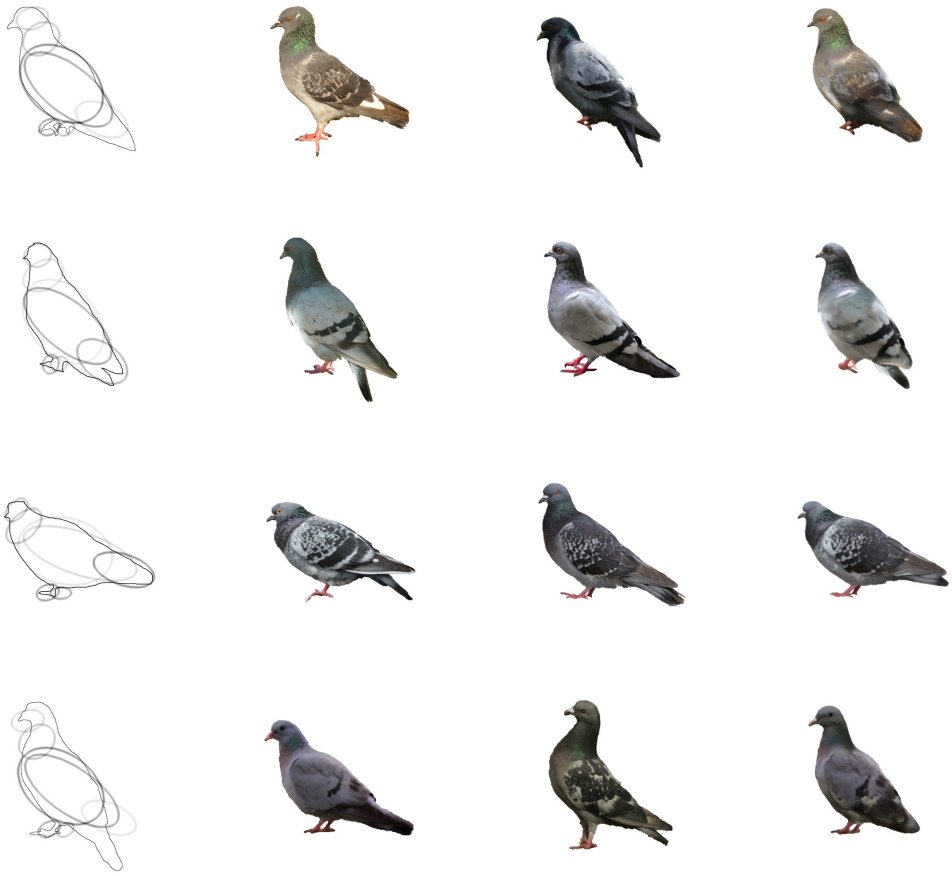


Figure C.1: From left to right: User's sketch; nearest neighbor from dataset; second nearest neighbor from dataset; synthesized result.

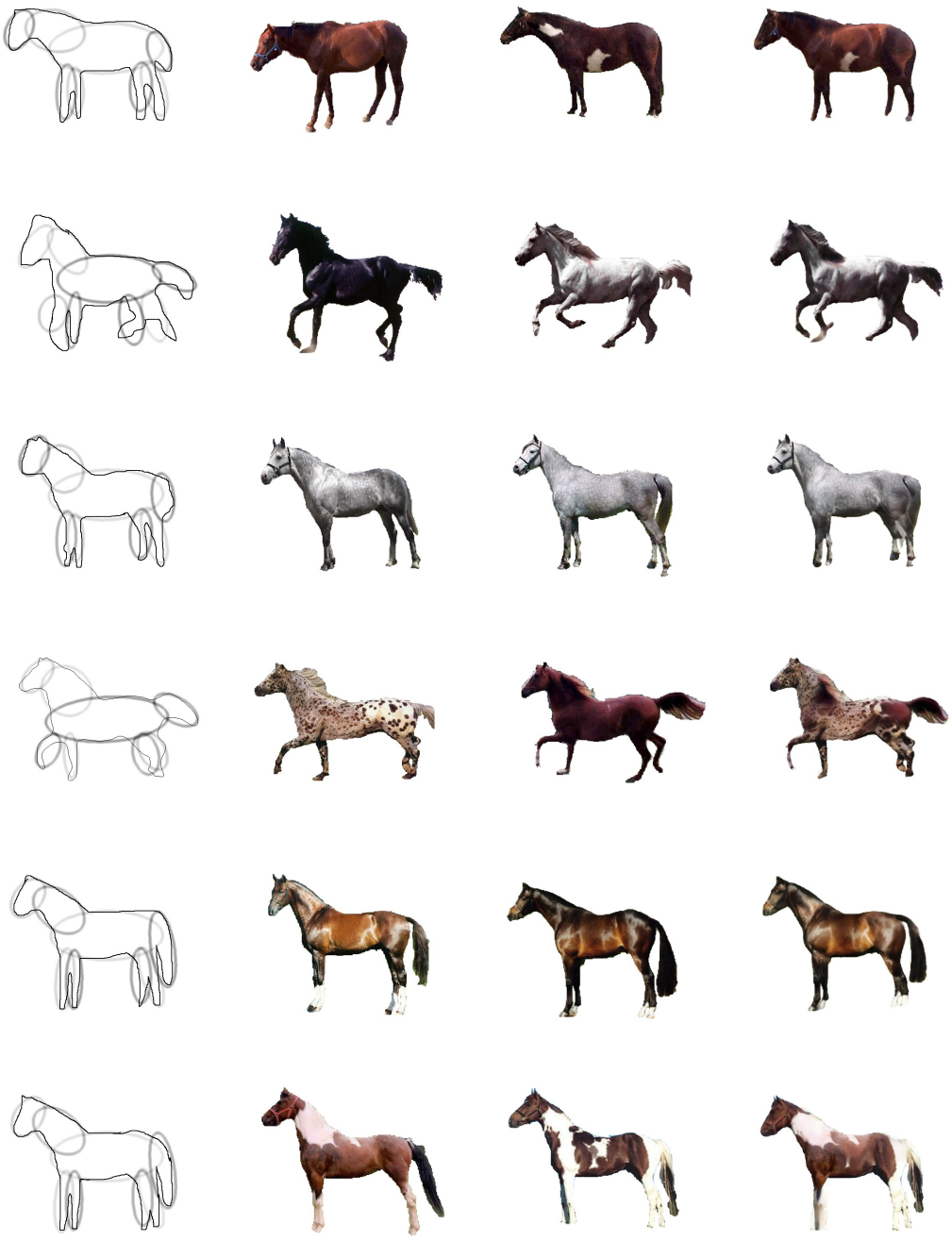


Figure C.2: From left to right: User's sketch; nearest neighbor from dataset; second nearest neighbor from dataset; synthesized result.

Figure C.3: From left to right: User's sketch; nearest neighbor from dataset; second nearest neighbor from dataset; synthesized result.

Appendix D

User Study Results

Sections D.1 and D.1 report user responses in the two user studies designed to analyze the Interactive Sketch-Driven Image Synthesis system described in Chapter 3. The analysis of the user responses is summarized in tables 3.2, 3.3 and 3.4.

D.1 First User Study: User Responses

User #	I can draw well.	I've been trained to draw with masses.	I think that I would like to use this system frequently.	I found the system unnecessarily complex.	I thought the system was easy to use.	I think that I would need the support of a technical person to be able to use this system.
1	Strongly agree	Agree	Neither agree nor disagree	Disagree	Neither agree nor disagree	Disagree
2	Neither agree nor disagree	Strongly disagree	Neither agree nor disagree	Disagree	Disagree	Strongly disagree
3	Neither agree nor disagree	Disagree	Agree	Disagree	Neither agree nor disagree	Disagree
4	Agree	Strongly disagree	Agree	Disagree	Agree	Strongly disagree
5	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree	Disagree
6	Disagree	Disagree	Neither agree nor disagree	Neither agree nor disagree	Agree	Strongly disagree
7	Disagree	Strongly disagree	Disagree	Strongly disagree	Neither agree nor disagree	Strongly disagree
8	Strongly disagree	Strongly disagree	Agree	Strongly disagree	Strongly agree	Disagree
9	Neither agree nor disagree	Strongly disagree	Agree	Disagree	Strongly agree	Disagree
10	Neither agree nor disagree	Disagree	Disagree	Neither agree nor disagree	Strongly agree	Agree
11	Strongly disagree	Strongly disagree	Disagree	Agree	Neither agree nor disagree	Disagree
12	Neither agree nor disagree	Strongly disagree	Agree	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree
13	Disagree	Strongly disagree	Agree	Disagree	Neither agree nor disagree	Neither agree nor disagree
14	Neither agree nor disagree	Agree	Neither agree nor disagree	Disagree	Agree	Disagree
15	Agree	Strongly disagree	Neither agree nor disagree	Neither agree nor disagree	Agree	Disagree
16	Disagree	Strongly disagree	Agree	Disagree	Agree	Strongly disagree
17	Neither agree nor disagree	Strongly disagree	Agree	Strongly disagree	Strongly agree	Strongly disagree
18	Agree	Strongly disagree	Neither agree nor disagree	Disagree	Agree	Strongly disagree

User #	I found the various functions in this system were well integrated.	I thought there was too much inconsistency in this system.	I would imagine that most people would learn to use this system very quickly.	I found the system very cumbersome to use.	I felt very confident using the system.	I needed to learn a lot of things before I could get going with this system.
1	Disagree	Agree	Agree	Disagree	Neither agree nor disagree	Disagree
2	Agree	Disagree	Disagree	Neither agree nor disagree	Disagree	Disagree
3	Agree	Agree	Agree	Disagree	Disagree	Strongly disagree
4	Agree	Agree	Strongly agree	Strongly disagree	Agree	Strongly disagree
5	Neither agree nor disagree	Neither agree nor disagree	Agree	Neither agree nor disagree	Neither agree nor disagree	Disagree
6	Neither agree nor disagree	Disagree	Agree	Disagree	Neither agree nor disagree	Disagree
7	Strongly agree	Strongly disagree	Strongly agree	Agree	Agree	Strongly disagree
8	Strongly agree	Disagree	Strongly agree	Strongly disagree	Disagree	Neither agree nor disagree
9	Strongly agree	Disagree	Agree	Disagree	Agree	Strongly disagree
10	Agree	Disagree	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree	Neither agree nor disagree
11	Strongly agree	Agree	Neither agree nor disagree	Agree	Agree	Strongly disagree
12	Strongly agree	Disagree	Agree	Neither agree nor disagree	Disagree	Disagree
13	Agree	Disagree	Agree	Neither agree nor disagree	Disagree	Strongly disagree
14	Agree	Agree	Agree	Disagree	Disagree	Disagree
15	Neither agree nor disagree	Neither agree nor disagree	Agree	Disagree	Neither agree nor disagree	Neither agree nor disagree
16	Strongly agree	Disagree	Agree	Disagree	Neither agree nor disagree	Disagree
17	Strongly agree	Strongly disagree	Strongly agree	Strongly disagree	Strongly agree	Strongly disagree
18	Agree	Strongly disagree	Agree	Disagree	Agree	Disagree

User #	I found the tools of the system worked well together.	The ellipse position feedback was useful.	The silhouette feedback was useful.	The fast "NN" preview image was useful.	The coarse preview of the generated image with the color choices was useful.	I was able to draw the specifications as I desired.
1	Neither agree nor disagree	Disagree	Agree	Neither agree nor disagree	Agree	Neither agree nor disagree
2	Agree	Agree	Disagree	Agree	Agree	Disagree
3	Agree	Agree	Strongly agree	Strongly agree	Agree	Neither agree nor disagree
4	Agree	Agree	Strongly agree	Agree	Strongly agree	Agree
5	Neither agree nor disagree	Disagree	Agree	Agree	Neither agree nor disagree	Neither agree nor disagree
6	Agree	Agree	Agree	Strongly agree	Neither agree nor disagree	Agree
7	Agree	Agree	Strongly agree	Agree	Strongly agree	Agree
8	Agree	Agree	Strongly agree	Neither agree nor disagree	Agree	Agree
9	Strongly agree	Agree	Strongly agree	Agree	Strongly agree	Agree
10	Agree	Disagree	Agree	Strongly agree	Strongly agree	Agree
11	Strongly agree	Disagree	Agree	Agree	Agree	Neither agree nor disagree
12	Strongly agree	Disagree	Strongly agree	Strongly agree	Agree	Disagree
13	Agree	Agree	Agree	Neither agree nor disagree	Agree	Agree
14	Neither agree nor disagree	Agree	Agree	Agree	Strongly agree	Neither agree nor disagree
15	Neither agree nor disagree	Agree	Neither agree nor disagree	Strongly agree	Agree	Strongly agree
16	Strongly agree	Agree	Strongly agree	Agree	Agree	Agree
17	Strongly agree	Strongly agree	Strongly agree	Strongly agree	Strongly agree	Agree
18	Agree	Neither agree nor disagree	Strongly agree	Agree	Strongly agree	Agree

User #	The generated image from Assignment 1 closely matches my specifications.	Which of the systems was the easiest to use?	"Given 3 images (ask surveyor to view them), which one is closest in terms of pose?"	"Given 3 images (ask surveyor to view them), which one is closest in terms of color?"	"Given 3 images (ask surveyor to view them), which one resembles the target image the most?"
1	Agree	System 2	The nearest neighbour found by the user	The nearest neighbour found by the system	The nearest neighbour found by the user
2	Agree	System 1	The nearest neighbour found by the user	The generated image	The generated image
3	Agree	System 2	The generated image	The nearest neighbour found by the user	The generated image
4	Agree	System 1	The nearest neighbour found by the user	The nearest neighbour found by the user	The nearest neighbour found by the user
5	Disagree	System 2	The nearest neighbour found by the user	The generated image	The nearest neighbour found by the user
6	Agree	System 1	The nearest neighbour found by the system	The nearest neighbour found by the user	The nearest neighbour found by the user
7	Disagree	System 2	The nearest neighbour found by the system	The nearest neighbour found by the user	The generated image
8	Agree	System 1	The nearest neighbour found by the user	The nearest neighbour found by the user	The nearest neighbour found by the user
9	Agree	System 2	The generated image	The nearest neighbour found by the user	The generated image
10	Strongly agree	System 2	The generated image	The nearest neighbour found by the user	The generated image
11	Agree	System 2	The nearest neighbour found by the user	The nearest neighbour found by the user	The nearest neighbour found by the user
12	Neither agree nor disagree	System 2	The nearest neighbour found by the system	The nearest neighbour found by the user	The nearest neighbour found by the user
13	Strongly agree	System 1	The generated image	The nearest neighbour found by the user	The nearest neighbour found by the user
14	Strongly agree	System 1	The nearest neighbour found by the user	The generated image	The nearest neighbour found by the user
15	Agree	System 1	The nearest neighbour found by the user	The nearest neighbour found by the user	The nearest neighbour found by the user
16	Agree	System 1	The nearest neighbour found by the user	The generated image	The generated image
17	Strongly agree	System 1	The nearest neighbour found by the user	The nearest neighbour found by the user	The nearest neighbour found by the user
18	Strongly agree	System 2	The generated image	The generated image	The generated image

D.2 Second User Study: User Responses

User #	The ellipse position feedback was useful.	The silhouette feedback was useful.	The fast "NN" preview image was useful.	The coarse preview of the generated image with the color choices was useful.	I was able to draw the specifications as I desired.	The generated image from Task 1 closely matches my specifications.	Which of the tasks was the easiest to perform?	When specifying pose, ellipses were useful.
1	Strongly agree	Strongly agree	Agree	Strongly agree	Strongly agree	Agree	System 1	Strongly agree
2	Agree	Agree	Agree	Agree	Strongly agree	Strongly agree	System 2	Agree
3	Agree	Disagree	Neither agree nor disagree	Agree	Neither agree nor disagree	Neither agree nor disagree	System 3	Agree
4	Strongly agree	Strongly agree	Strongly agree	Strongly agree	Agree	Neither agree nor disagree	System 4	Strongly agree
5	Neither agree nor disagree	Agree	Strongly disagree	Agree	Disagree	Disagree	System 5	Agree
6	Agree	Agree	Neither agree nor disagree	Strongly agree	Agree	Agree	System 6	Agree

D.3 Intermediate Results of Image Synthesis

Figure D.1: Detailed partial images of the horses generated with 9×9 patches. Each image corresponds to image T_{partial} used to generate results in Figure 5.16.

Figure D.2: Blending of 3×3 patches found by minimizing $\mathcal{L}[T, T_{\text{partial}}]$ for the random samples of horses in Figure 5.15.